Tampere University

Anna Mercader Ardèvol

OPTIMIZATION OF STEEL FRAMES USING TWO-PHASE APPROACH

Faculty of Built Environment
Master's thesis
May 2019

ABSTRACT

Anna Mercader Ardèvol: Optimization of steel frames using two-phase approach
Master's thesis
Tampere University
Civil Engineering
May 2019

The goal of this thesis was to develop a program able to solve steel frame optimization problems using a two-phase approach. This optimization problem consists of minimizing the weight of a steel frame ensuring that the solution found satisfies all the strength and stability criteria establish in Eurocode 3.

The two-phase approach consists of first finding a continuous solution and after it, using this first solution to select the closest standardized profiles and find among them the optimum discrete solution. This optimization method was implemented using Python and checked with Ftool that all the structural calculations were correct.

Two different formulations had been written following the same structure but with a big difference, the first one has only one design variable (h) and the second has 5 (h, b, t_f, t_w, r). As the first formulation has only one design variable, the correlation between h and all the other design variables needed to be found. To be able to approximate all the other variables it is necessary to know which profile family is each member of the frame, for this reason in the first formulation all columns are HEA profiles and beams IPE. However, in the second problem, all members can be from any of the I families.

These two different formulations had been done to compare if the results obtained using all the design variables improve significantly and how much the computation time increased. This way, it was possible to compare if it was worth it to use all the variables. Moreover, three different optimization algorithms were used in the first phase to check which one was better.

Keywords: optimization, steel frame, weight minimization, two-phase approach

PREFACE

This thesis was done in the research group Metal and lightweight structures in the Civil Engineering Department of the University of Tampere from August 2018 to April 2019.

I am grateful to Professor Sami Pajunen who help me finding a suitable topic for my master thesis and a place in the research group to do it.

I would like to thanks all the support received from my supervisor Kristo Mela who has been advising me during my entire thesis and providing me with helpful literature. I also want to thanks Jaakao Huusko who help me when I was having trouble with some implementations in Python and to all the other persons in the research group that give me all their support.

Finally, I express my gratitude to my family and my friends who have always been encouraging me during all the process of writing this thesis and motivating me when I was having problems with some parts.

Tampere, 3 May 2019

Anna Mercader Ardèvol

CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF SYMBOLS AND ABBREVIATIONS

Abbreviations

COBYLA	Constrained optimization by linear approximation
GA	Genetic algorithm
LP	Linear programming
MILP	mixed integer linear programming
PSO	Particle swarm optimization
SLSQP	Sequential least-squares quadratic programming
TSP	Travelling salesman problem

Symbols

A	Cross- section area
A_v	Shear area
b	Width of a cross-section
c	Longitude of the flange or web
E	Young modulus
f_y	Elastic limit
G	Shear modulus
h	Height of a cross-section
I	Moment of inertia
l	Member's length
N_{cr}	Elastic critical force for flexural buckling
$N_{b,Rd}$	Design buckling resistance
$N_{c,Rd}$	Design resistance to normal force for uniform compression
N_{Ed}	Design value of the axial force
M_{cr}	Elastic critical moment for lateral torsional buckling
$M_{b,Rd}$	Design buckling resistance moment
$M_{c,Rd}$	Design resistance for bending
M_{Ed}	Design bending moment
$M_{el,Rd}$	Design elastic resistance for bending
$M_{pl,Rd}$	Design plastic resistance for bending
r	Radius between flange and web union
t	Thickness
t_f	Flange thickness
t_w	Web thickness
$V_{c,Rd}$	Design shear resistance
V_{Ed}	Design shear force
$V_{pl,Rd}$	Design plastic shear resistance
$W_{el,min}$	Minimum elastic section modulus
W_{pl}	Plastic section modulus
W_y	Section modulus
$y-y$	Axis of a cross-section
$z-z$	Axis of a cross-section
α	Portion of a part of a cross-section in compression
α	Imperfection coefficient for buckling curves
α_{LT}	Imperfection coefficient for lateral buckling curves
ρ	Steel density
ν	Poisson's ratio in elastic stage

Ψ	Stress ratio
ε	Coefficient depending of f_y
γ_{M0}	Partial safety factor for the resistance of cross-section
γ_{M1}	Partial safety factor for the resistance of members to buckling
χ	Reduction factor for relevant buckling mode
χ_{LT}	Reduction factor for lateral buckling
Φ	Value to determine the reduction factor χ
Φ_{LT}	Value to determine the reduction factor χ_{LT}
λ	Non-dimensional slenderness for buckling
λ_{LT}	Non-dimensional slenderness for lateral torsional buckling

1. INTRODUCTION

1.1 Background

Steel frames are commonly used in structural engineering for building any kind of edification such as residential houses or industrial buildings. However, the election of profiles made is not always the best one as far as it refers to the minimum weight of the frame.

Sometimes, if a different selection of profiles had been used, the weight of the structure would have been lower. To solve this, structural optimization is used and it consists on finding the best feasible design of the structure using less material but ensuring that it can resist all the loads applied and satisfies all the constraints defined by the standards like Eurocode 3.

Figure 1 shows how the discretization of a building's steel frame is made to be able to do the structural optimization.

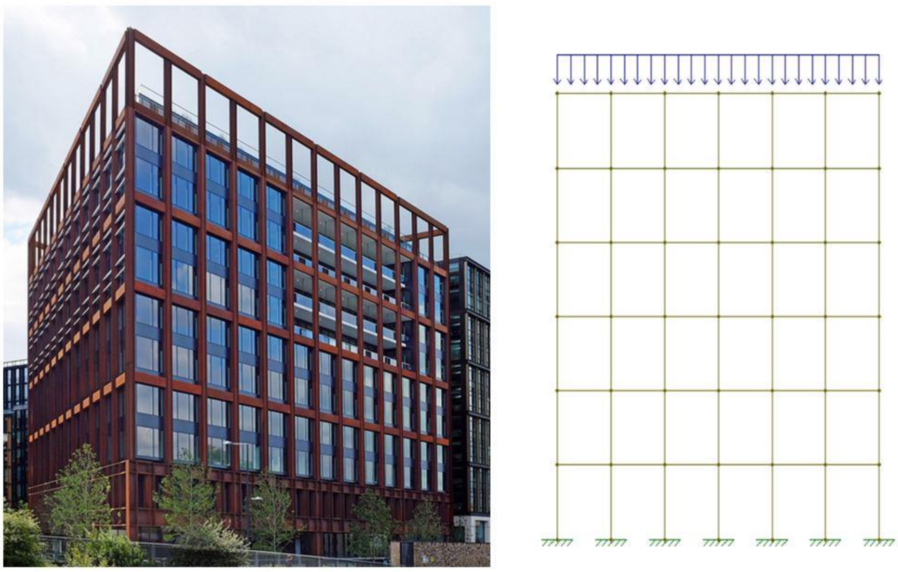

Figure 1. *Building discretization*

Structural optimization literature started in 1890 with Maxwell's work [1] and followed by Mitchell in 1904 [2]. During World War 2, the weight optimization of common aircraft elements like columns with compression loads applied began to be developed.

In the 1950's, the space programs increased the demand for lightweight structures and provided the means to develop new design approaches. Furthermore, the finite element method was offering the designers a powerful technic for the analysis of complex structures, which helped to make important advances in structural optimization [3].

In 1960, Schmit [4] was the first offering an extensive report on how to use mathematical programming techniques to solve nonlinear inequality constrained problems to design elastic structures with multiple load conditions. He introduced the idea of combining finite element structural analysis and nonlinear mathematical programming and showed that it was a feasible idea [5]. In the 1960's Prager and coworkers [6][7] presented an alternative approach, called Optimality Criteria which was quite intuitive and a more effective design tool [5].

The need for discrete variable structural optimization has been recognized since 1968 with the work of Toakley [8]. However, since the continuous variable optimization algorithms were not fully developed yet, the emphasis focused on developing algorithms for those problems. At that time, the integer variable linear programming (LP) and the branch and bound methods had been developed for general discrete optimization and Toakley applied a few of them for optimum design of plastic and elastic structures [9]. In addition, Cella and Logcher solved in 1971 the nonlinear problem of designing trusses subjected to stress constraints using the branch and bound method [10].

In the last 20 years, the quantity of research articles using gradient-free algorithms for solving structural optimization problems has increased. In particular, evolutionary algorithms have risen as an important technique for structural optimization [11][12][13].

The ant colony optimization has been also used recently, in 2004 Camp and Bichon adapted the ant colony optimization to the design of space trusses by mapping the structural design problem into a modified traveling salesman problem (TSP) [14] and in 2005 for the optimization of steel frames [15].

Another methodology used for discrete optimization is the two-phase approach. It was developed in 1988 by Hager and Balling [16] and was a technique for selecting optimum members in steel frames. It consisted of first solving the continuous problem and with the optimum solution found, select a neighborhood of discrete profiles and finally use the branch and bound method with the selected profiles to find the discrete solution.

The two-phase approach has continued being studied and some of the algorithms that are used in each phase are explained in the review written by Arora and Huang in 1996 [17] and in 1997 [18].

Nowadays, there are three different strategies to formulate structural optimization: sizing, shape and topology optimization and the two-phase approach can be used to solve them [19][20].

Sizing optimization minimizes the weight of a structure under behavioral constraints of stress and displacements optimizing the different parameters of the cross-section of the different members of the frame. As the cross-section of the profiles has manufacturing restrictions, the cross-section parameters are discrete and need to correspond to a profile available in the market.

Shape optimization improves the performance of the structure by modifying the geometry. This optimization uses the nodes of the frame as design variables to modify and achieve the best solution.

Topology optimization defines the type of structure suited to satisfy the operating conditions for the problem and makes a rational arrangement of the available material. This means that the material inefficiently used is eliminated gradually, checking every time that the frame remains functional [21].

In Figure 2, an example of each optimization strategy is shown.

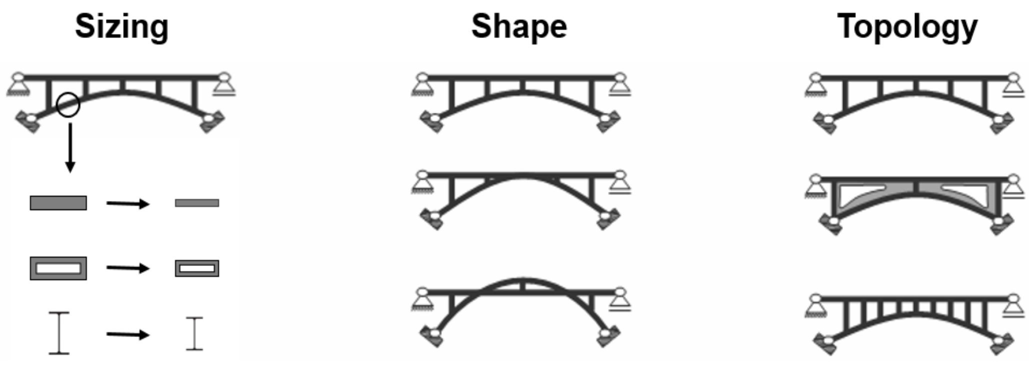

Figure 2. *Sizing, shape and optimization problems*

This thesis focuses on sizing optimization assuming that the shape and topology of the frame are already fixed. The goal is to search between all type I profiles available in the market and find the best profile for each member of the frame to guarantee the minimum weight. The discrete profiles among which the optimum is going to be searched can be IPE, HEA, HEAA, HEB, HEC and HEM, view Figure 3.

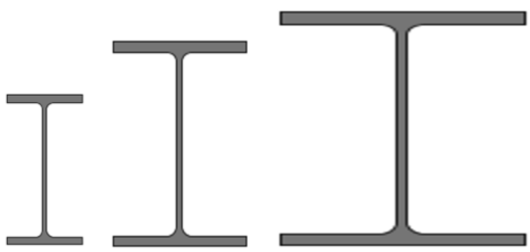

Figure 3. *Different I profile families*

The most common profiles used are HEA profiles for the columns and IPE for beams. However, this is not always the optimum solution, for this reason, in the thesis each member of the frame can be from a different family of profiles allowing to find better solutions.

1.2 Aim and scope of the study

The main objective of the study is the minimization of the weight of steel frames using a two-phase approach. The first and second phases solve the continuous and discrete problems respectively and are explained in chapter 5.

In the first phase, the result obtained is the optimum solution, however, it is not feasible because as it is a continuous solution the profiles do not match any commercial profile. For this reason, it is necessary to use a discrete approach to find the closest standardized profile to the optimum solution.

This way is going to be possible to find which type of profile is better to use making sure that it resists all the loads applied, that can be uniform or punctual loads, and respecting all the constraints and restrictions establish in the Eurocode 3 [22].

Different optimization algorithms are used and compare to study which one gets better results and requires less computation time. All these optimization algorithms are implemented using Python and the results saved automatically in excel documents to make an easier analysis of the obtained data.

To make sure that all the calculations were programmed correctly, the reactions and displacements obtained as a result of applying different loads to a studied frame were checked with Ftool software.

The aim of this thesis was to answer the following questions:

1) Is the two-phase approach an efficient and reliable tool to solve structural optimization problems?
2) Which are the most suitable algorithms to solve the optimization problem?
3) Is the computation time for solving the optimization problem with any family of profiles significantly longer than doing it using a fixed profile family?

These questions are going to be answered making a program with Python able to solve the structural optimization problem using the two-phase approach. The continuous part is going to be solved with 3 different optimizer algorithms and the discrete solution with a genetic algorithm (GA).

This program will have two different formulations, the first one will have only 1 design variable per member and the family type of profile for beams and columns will have been decided in advanced. The second formulation will have the 5 cross-section parameters as design variables allowing the frame members to be from any type I profile family.

Three different case studies, each one more complex, are going to be designed and solve with the program to analyze the results obtained.

2. OPTIMIZATION BASIS

Optimization consists of maximizing or minimizing an objective function while satisfying all constraints needed on the problem. This objective function has different design variables, which are the parameters that are needed to be found to reach the optimum result, and it can be subject to 2 different types of constraints: inequality constraints or equality constraints [23].

2.1 Basic concepts

Continuity

A function $f(x)$ is continuous in the point a if the following conditions are true:

1) $f(a)$ is defined, a exists in the domain of f.
2) $\lim_{x \to a} f(x)$ exists for x in the domain of f.
3) $f(a) = \lim_{x \to a} f(x)$.

Differentiability

A function that is continuous at every point and the derivative of each point of the domain exists. As a consequence, a tangent line must be able to be drawn at every point of the function, view Figure 4.

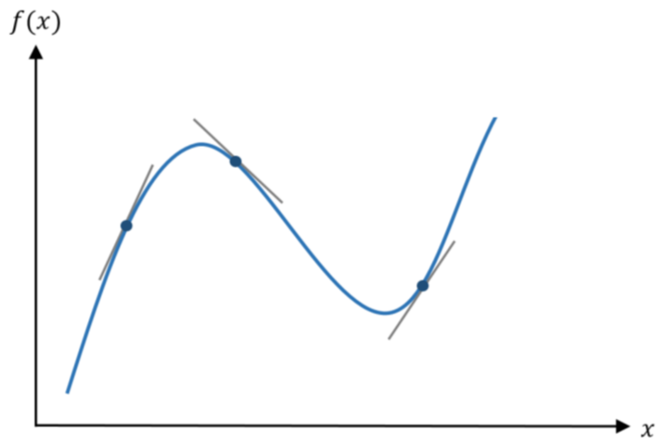

Figure 4. *Differentiable function*

A function f is differentiable at a if:

1) f is continuous at a.
2) The derivative $f'(a) = \lim_{h \to 0} \frac{f(a+h)-f(a)}{h}$ exists.

Local optimum

Local optimum is an optimum point comparing to the feasible solutions in its neighborhood. The points that are far from the local optimum might be a better solution but they do not play any role in the definition of the local optimum [24].

Global optimum

Global optimum is a point in the feasible region whose objective value is better than any other point.

Figure 5 shows the difference between a local and a global optimum, precisely between a local and global minimum.

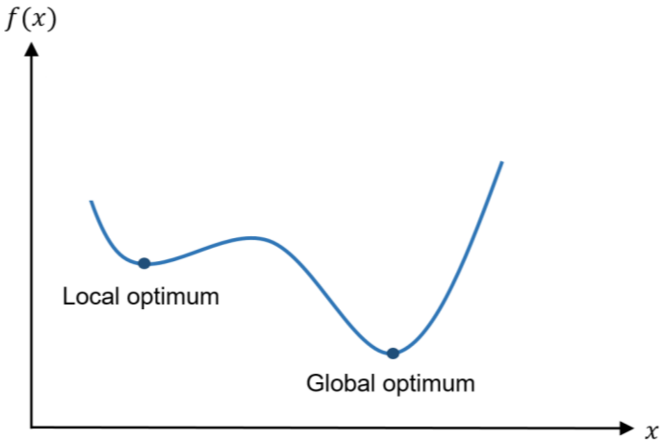

Figure 5. *Local and global optimum*

Convex function

A function is convex if any two points on the graph's function can be joint with a line and this line lies above or on the graph view Figure 6. It is also a continuous function.

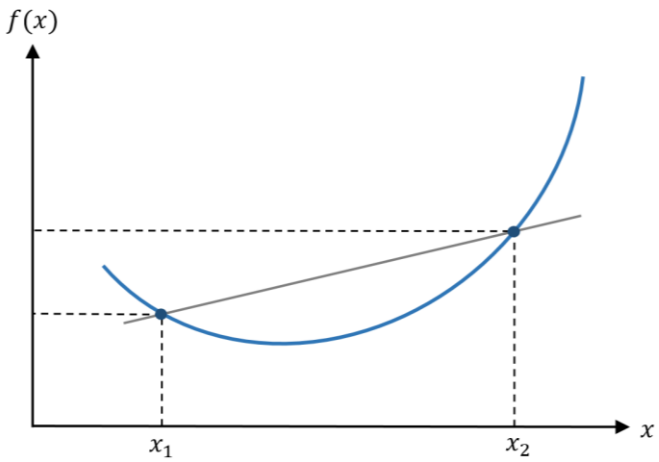

Figure 6. *Convex function*

Gradient

The gradient is a vector of partial derivatives of first-order. Points the direction of greatest increase of a function and its value is zero at a local minimum or maximum.

$$\nabla f = \begin{bmatrix} \frac{\partial f}{\partial x_1} \\ \frac{\partial f}{\partial x_2} \\ \vdots \\ \frac{\partial f}{\partial x_\mathrm{n}} \end{bmatrix} \tag{1}$$

Hessian matrix

Square symmetric matrix of second-order partial derivatives that describes the local curvature of a function and helps to simplify optimization procedures and finding a solution for problems with a large number of variables [25].

$$H = \nabla^2 f = \begin{bmatrix} \frac{\partial^2 f}{\partial x_1^2} & \frac{\partial^2 f}{\partial x_1 \partial x_2} & \cdots & \frac{\partial^2 f}{\partial x_1 \partial x_n} \\ \frac{\partial^2 f}{\partial x_2 \partial x_1} & \frac{\partial^2 f}{\partial x_2 \partial x_2} & \cdots & \frac{\partial^2 f}{\partial x_2 \partial x_n} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial^2 f}{\partial x_n \partial x_1} & \frac{\partial^2 f}{\partial x_n \partial x_2} & \cdots & \frac{\partial^2 f}{\partial x_1^2} \end{bmatrix} \tag{2}$$

2.2 Formulation of a general optimization problem

The general formulation of a constrained minimization problem is:

$$\begin{array}{lll} \text{Minimize} & f(x) & \\ \text{subject to} & g_i(x) \leq 0 & i = 1, \dots, m \\ \text{and} & h_j(x) = 0 & j = 1, \dots, l \end{array} \tag{3}$$

Each element of this problem is explained below [26][27].

Design variable

Values whose numerical quantities need to be chosen during the optimization.

$$x = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} \tag{4}$$

Objective function

The objective function is the desired function to minimize or maximize which has x_n design variables.

This function is the criterion that allows comparing the different solutions and selecting the best one.

$$f(x) = f(x_1, x_2, \dots, x_n) \tag{5}$$

Constraints

Restrictions fixed by the environment and processes that need to be satisfied in order to produce an acceptable solution. These restrictions describe dependencies among decision variables.

- Inequality constraints: include an explicit upper and lower bounds of the design variables.

$$g_i(x) \leq 0 \quad i = 1, \dots, m \tag{6}$$

- Equality constraints: the number of equality constraints must be less than n.

$$h_j(x) = 0 \quad j = 1, \dots, l \tag{7}$$

3. STEEL FRAME OPTIMIZATION PROBLEM

The problem to be solved is a steel frame with a variable number of bays, storeys and different loads applied. This frame can be symmetric or not and the information required is: number of bays and storeys, bay length [m], storey height [m] and the loads applied which can be uniform loads on beam or columns [kN/m] or point load [N], see Figure 7.

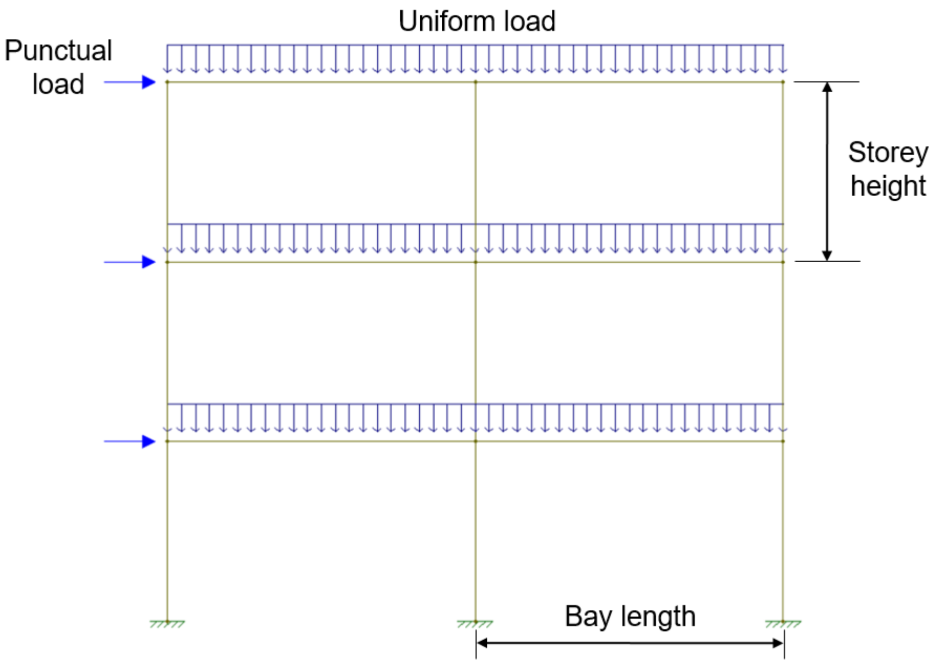

Figure 7. *Steel frame to study*

In case the frame is not symmetric, it is possible to design it entering the coordinates of the origin and end nodes of each member.

For both cases is also required to establish how many elements per member, and it has been decided to have 4 elements per member in the cases analyzed.

The other information needed by default is set as shown in Table 1.

Table 1. *Problem default parameters*

Initial beam	**Initial column**	**Young modulus E [N/mm^2]**	**Shear modulus G [N/mm^2]**	**Elastic limit f_y [N/mm^2]**	**Poisson's ratio ν [N/mm^2]**	**Steel density ρ [kg/mm^3]**
IPE 400	HEA 400	210 x 10^3	81 x 10^3	355	0.3	7850 x 10^{-9}

3.1 Objective function

The objective function to minimize is the weight of the frame because the minimum weight implies less material and as a result, less cost. As each frame member can be different, the objective function is going to be calculated for each one using the following formula.

$$f(x) = \sum_{i=1}^{n} A_i \, l_i \rho_i \tag{8}$$

Where n is the number of frame members, A_i is the cross-section area, l_i the length of the member and ρ_i the material density. Specifically in this thesis, all members are I profiles and in this case, the cross-section area is calculated:

$$A_i = 2t_{f,i}b_i \; + \; \left(h - 2t_{f,i}\right)t_{w,i} \; + \; (4 - \pi)r_i^2 \tag{9}$$

The design variables needed to calculate the area of the profile are explained in the next section.

3.2 Design variables

This thesis is taking into account different types of profiles, which are: IPE, HEA, HEAA, HEB, HEC, HEM [28].

To define properly all the cross-sections, 5 different parameters are needed: profile height (h), profile width (b), flange thickness (t_f), web thickness (t_w) and radius between flange and web union (r), view Figure 8:

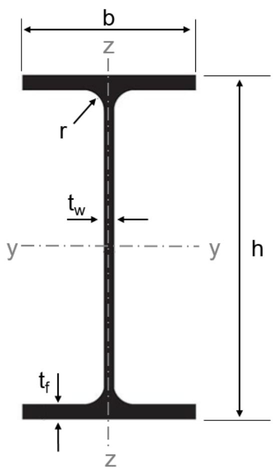

Figure 8. *Cross-section parameters*

These cross-section parameters are the design variables of the optimization problem. In this thesis, two different formulations have been developed and in the first one is only used the h variable (10) and in the second all the design variables (11).

$$X = [h_1, h_2 \dots h_n] \tag{10}$$

$$X = [h_1, b_1, t_{f,1}, t_{w,1}, r_1 \dots h_n, b_n, t_{f,n}, t_{w,n}, r_n] \tag{11}$$

The first formulation only uses one design variable for each frame member because a correlation between all these parameters has been done and as a result, it is possible to approximate b, t_f, t_w, r knowing only the h and the type of profile.

Table 2 shows all the correlation between h and the other design variables for each profile family. This correlation was found using all the cross-section parameters of the standardized profiles [28] and plotting them on Excel to obtain each correlation.

Table 2. *Correlation between cross-section parameters per families*

	Correlation between parameters	
IPE	$b = 0.3486h + 30.178$	
	$t_f = 0.0256h + 3.2831$	
	$t_w = 0.0155h + 2.4921$	
	$r = 0.0155h + 2.4921$	
HEA	$b = 1.0323h + 2.5368$ $b = 300$	$if\ h \leq 270$ $if\ h > 270$
	$t_f = 0.0294h + 5.7651$	
	$t_w = 0.014h + 4.2949$	
	$r = 0.0781h + 2.9206$ $r = 27$ $r = 30$	$if\ h \leq 270$ $if\ 270 < h \leq 690$ $if\ h > 690$
HEAA	$b = 1.0403h + 6.2902$ $b = 300$	$if\ h \leq 264$ $if\ h > 264$
	$t_f = 0.0179h + 4.9342$	
	$t_w = 0.0144\text{h} + 3.2122$	
	$r = 0.0786h + 3.221$ $r = 27$ $r = 30$	$if\ h \leq 270$ $if\ 270 < h \leq 670$ $if\ h > 670$
HEB	$b = h$ $b = 300$	$if\ h \leq 280$ $if\ h > 280$
	$t_f = 0.0308h + 9.5736$	
	$t_w = 0.0147h + 6.2014$	
	$r = 0.0755h + 2.7636$ $r = 27$ $r = 30$	$if\ h \leq 280$ $if\ 280 < h \leq 700$ $if\ h > 700$

HEC	$b = 0.9256h + 5.593$	
	$t_f = 0.0669h + 6.7987$	
	$t_w = 0.0317h + 5.4267$	
	$r = 0.0747h + 2.0905$	
HEM	$b = 0.9344h - 2.8966$	$if \ h \leq 340$
	$b = -0.0112h + 312.41$	$if \ h > 340$
	$t_f = 0.0824h + 8.5346$	$if \ h \leq 340$
	$t_f = 40$	$if \ h > 340$
	$t_w = 0.0392h + 6.8259$	$if \ h \leq 340$
	$t_w = 21$	$if \ h > 340$
	$r = 0.0732h + 1.5707$	$if \ h \leq 340$
	$r = 27$	$if \ 340 < h \leq 716$
	$r = 30$	$if \ h > 716$

In the graphics, this correlation is found and also the coefficient of determination R^2, which is a statistical parameter that indicates how well is predicted the dependent variables (b, t_f, t_w, r) from the dependent (h). R^2 coefficient has a value between 0 and 1 and the higher it is, the better the regression model fits the data.

Figure 9 shows the correlation for IPE profiles, Figure 10 for HEA, Figure 11 for HEAA, Figure 12 for HEB, Figure 13 for HEC and Figure 14 for HEM.

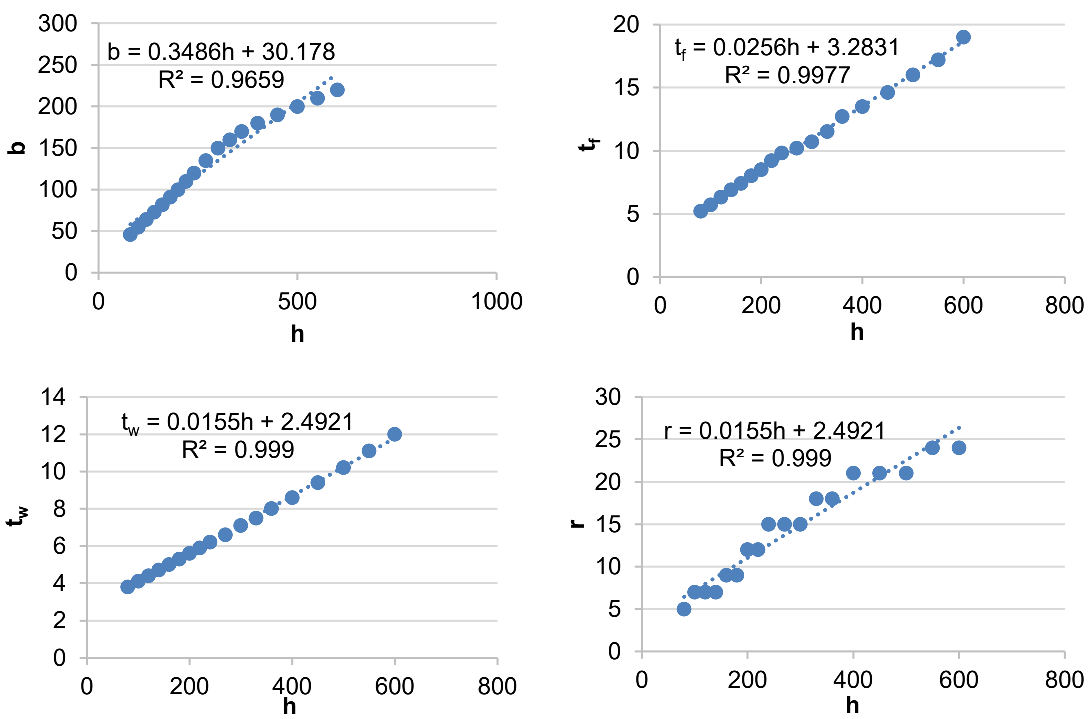

Figure 9. *Correlation between cross-section parameters for IPE profiles*

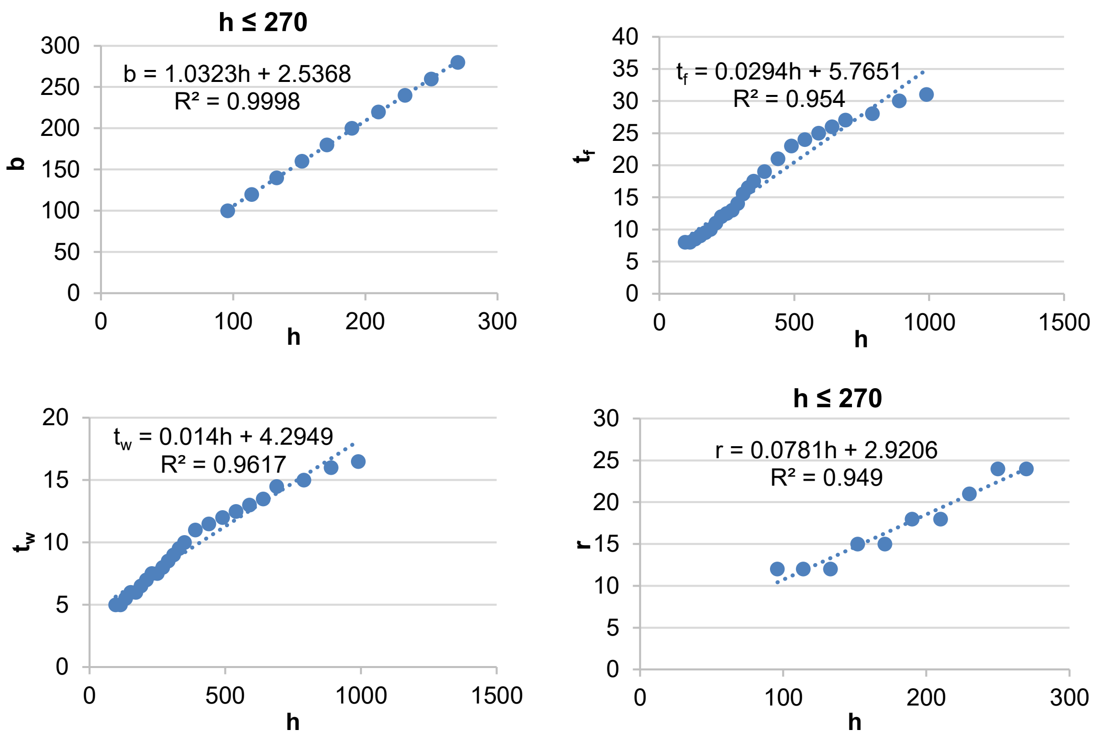

Figure 10. *Correlation between cross-section parameters for HEA profiles*

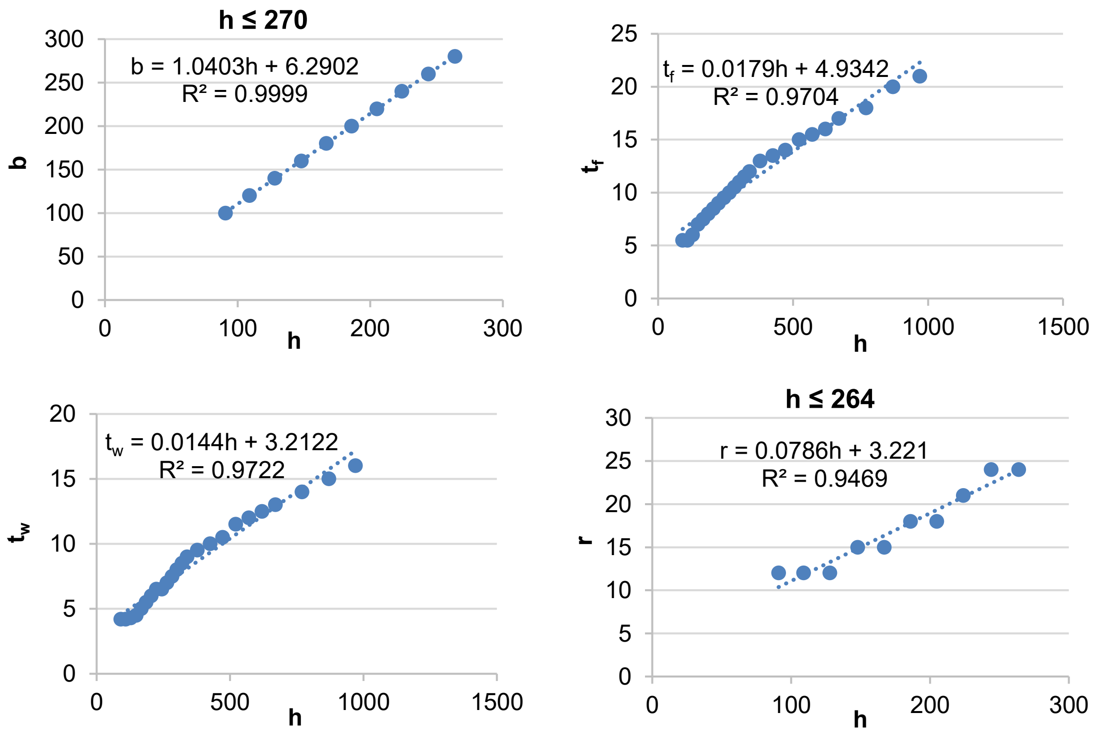

Figure 11. *Correlation between cross-section parameters for HEAA profiles*

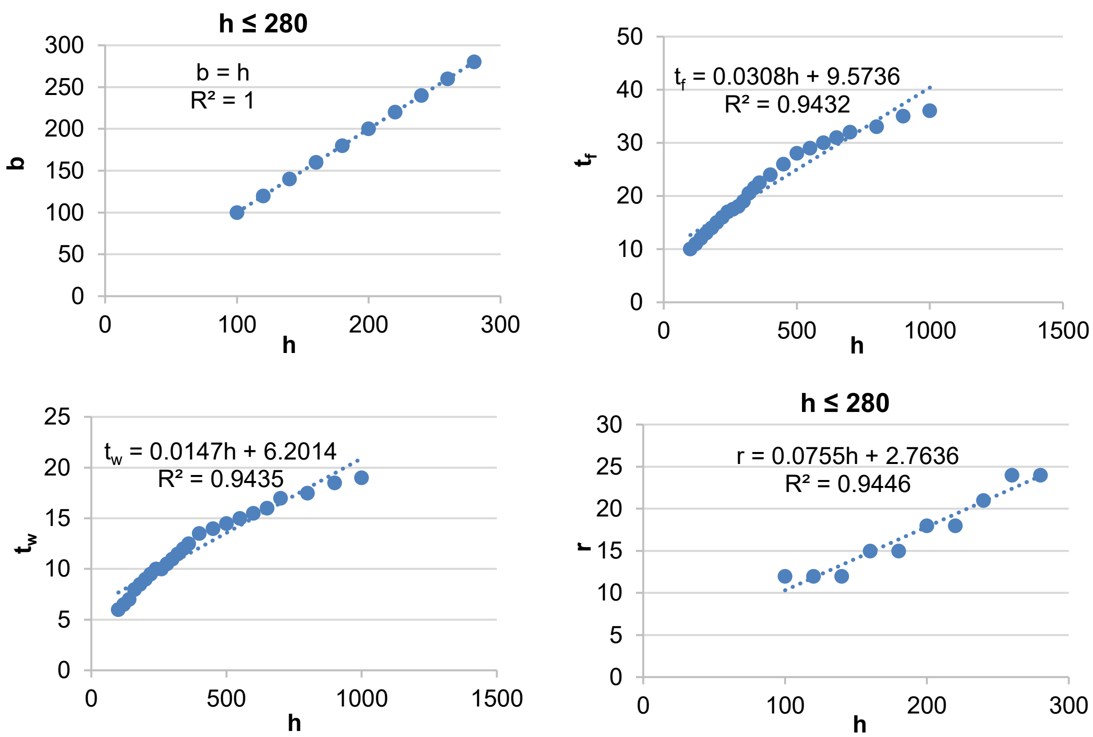

Figure 12. *Correlation between cross-section parameters for HEB profiles*

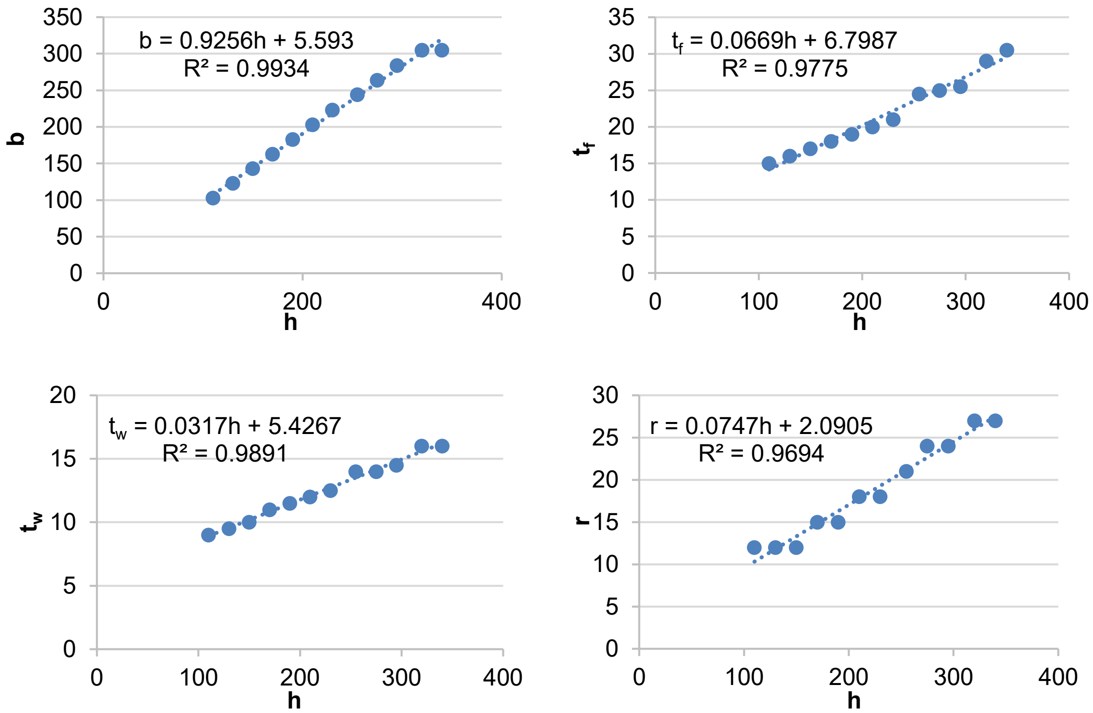

Figure 13. *Correlation between cross-section parameters for HEC profiles*

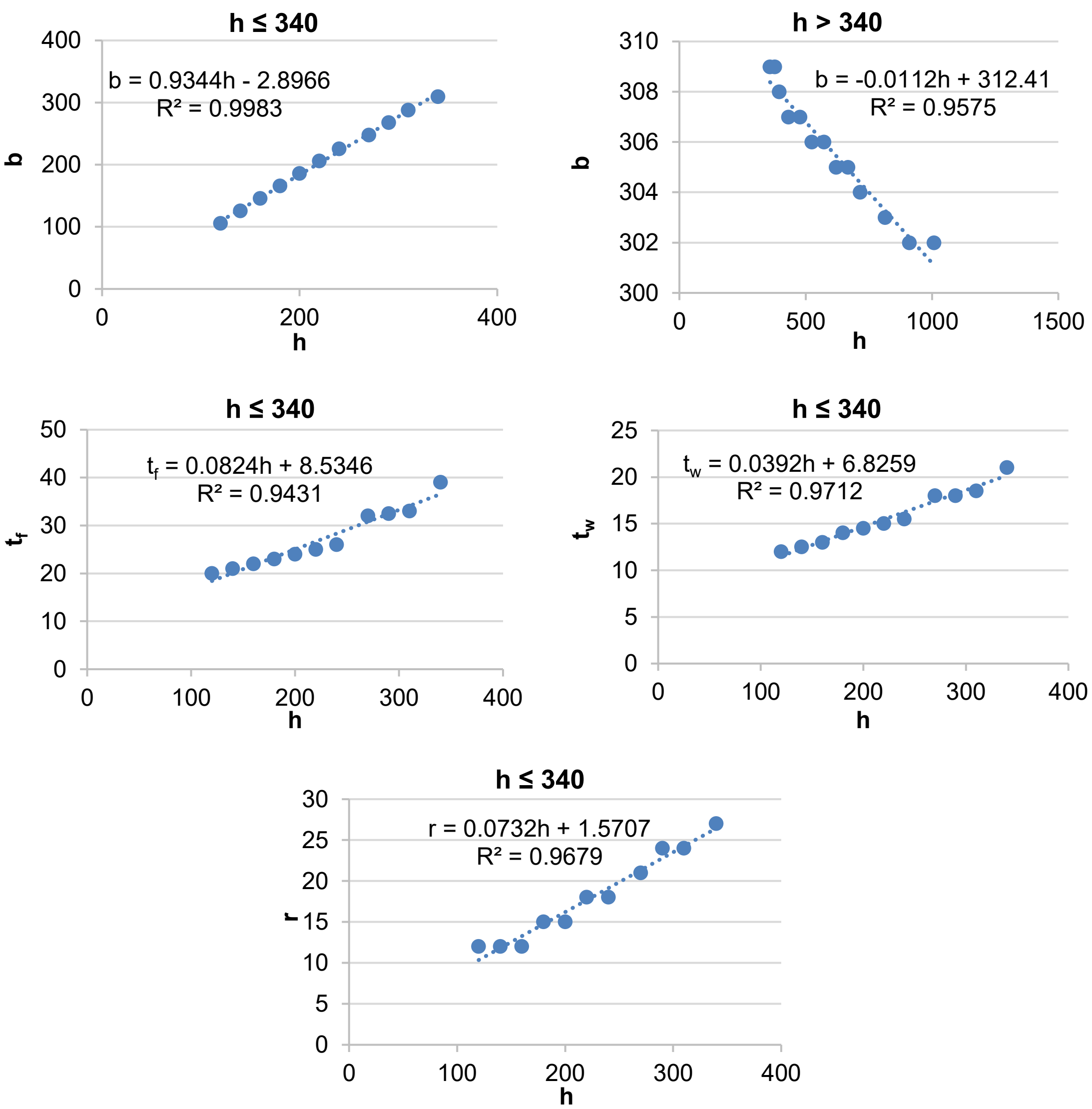

Figure 14. *Correlation between cross-section parameters for HEM profiles*

3.3 Constraints

According to EN 1993-1-1 steel members subjected to axial force and bending moment need to satisfy constraints of cross-section resistances, member stability and global stability to ensure that the structure resists all the forces applied.

To check that all these constraints are satisfied, first it is needed to identify the class of each member. The class depends on cross-section parameters and the forces applied to the section, if it is only compression, bending or a combination of these two and the program does this classification automatically using table 5.2 from EN 1993-1-1 [22].

After finding the class of each section, it is necessary to verify that the axial force, bending moment and shear force satisfy the formulas established by the Eurocode 3.

Compression

The design value of compression force N_{Ed} needs to satisfy:

$$\frac{N_{Ed}}{N_{c,Rd}} \leq 1 \tag{12}$$

Where the design resistance for uniform compression $N_{c,Rd}$ is:

$$N_{c,Rd} = \frac{Af_y}{\gamma_{M0}} \text{ class 1, 2 or 3} \tag{13}$$

And $\gamma_{M0} = 1,\ f_y = 355\ N/mm^2$.

Bending moment

The design bending moment M_{Ed} needs to satisfy:

$$\frac{M_{Ed}}{M_{c,Rd}} \leq 1 \tag{14}$$

Where the design resistance for bending $M_{c,Rd}$ is:

$$M_{c,Rd} = M_{pl,Rd} = \frac{W_{pl} f_y}{\gamma_{M0}} \text{ if it is class 1 or 2} \tag{15}$$

$$M_{c,Rd} = M_{el,Rd} = \frac{W_{el,min} f_y}{\gamma_{M0}} \text{ if it is class 3} \tag{16}$$

Shear force

The design value of shear force V_{Ed} needs to satisfy:

$$\frac{V_{Ed}}{V_{c,Rd}} \leq 1 \tag{17}$$

Where the design shear resistance $V_{c,Rd}$ is:

$$V_{c,Rd} = V_{pl,Rd} = \frac{A_v (f_y/\sqrt{3})}{\gamma_{M0}} \tag{18}$$

And shear area A_v is calculated:

$$A_v = A - 2bt_f + (t_w + 2r)t_f \tag{19}$$

Flexural buckling

If a member is compressed its buckling needs to be checked:

$$\frac{N_{Ed}}{N_{b,Rd}} \leq 1 \tag{20}$$

Where the design buckling resistance $N_{b,Rd}$ is:

$$N_{b,Rd} = \frac{\chi A f_y}{\gamma_{M1}} \text{ if it is class 1, 2 or 3} \tag{21}$$

The reduction factor χ, slenderness λ and Φ are:

$$\chi = \frac{1}{\Phi + \sqrt{\Phi^2 - \lambda^2}} \leq 1 \tag{22}$$

$$\Phi = 0.5[1 + \alpha(\lambda - 0.2) + \lambda^2] \tag{23}$$

$$\lambda = \sqrt{\frac{A f_y}{N_{cr}}} \tag{24}$$

The imperfection coefficient α has a different value depending on the buckling curve as shown in Table 3.

Table 3. *Imperfection coefficient for buckling curves*

Buckling curve	a_0	a	b	c	d
$\boldsymbol{\alpha}$	0.13	0.21	0.34	0.49	0.76

Lateral torsional buckling

The lateral torsional buckling has to be checked as follows:

$$\frac{M_{Ed}}{M_{b,Rd}} \leq 1 \tag{25}$$

Where the design buckling resistance moment $M_{b,Rd}$ is:

$$N_{b,Rd} = \frac{\chi_{LT} W_y f_y}{\gamma_{M1}} \tag{26}$$

The reduction factor χ_{LT}, slenderness λ_{LT} and Φ_{LT} are

$$\chi_{LT} = \frac{1}{\Phi_{LT} + \sqrt{\Phi_{LT}^2 - \lambda_{LT}^2}} \tag{27}$$

$$\Phi_{LT} = 0.5[1 + \alpha_{LT}(\lambda_{LT} - 0.2) + \lambda_{LT}^2] \tag{28}$$

$$\lambda_{LT} = \sqrt{\frac{W_y f_y}{M_{cr}}} \tag{29}$$

The imperfection coefficient α_{LT} has a different value depending on the buckling curve for lateral as shown in Table 4.

Table 4. *Imperfection coefficient for lateral torsional buckling curves*

Buckling curve	a	b	c	d
$\boldsymbol{\alpha_{LT}}$	0.21	0.34	0.49	0.76

Buckling curve

The selection of the buckling curve depends on the relation between h and b cross-section parameters and t_f value as shown in Table 5.

Table 5. *Buckling curve for each cross-section*

<table>
<tr><th></th><th></th><th>Buckling axis</th><th>Buckling curve</th></tr>
<tr><td rowspan="4">$\frac{h}{b} > 1.2$</td><td rowspan="2">$t_f \leq 40\ mm$</td><td>y-y</td><td>a</td></tr>
<tr><td>z-z</td><td>b</td></tr>
<tr><td rowspan="2">$t_f > 40\ mm$</td><td>y-y</td><td>b</td></tr>
<tr><td>z-z</td><td>c</td></tr>
<tr><td rowspan="4">$\frac{h}{b} \leq 1.2$</td><td rowspan="2">$t_f \leq 100\ mm$</td><td>y-y</td><td>b</td></tr>
<tr><td>z-z</td><td>c</td></tr>
<tr><td rowspan="2">$t_f > 100\ mm$</td><td>y-y</td><td>d</td></tr>
<tr><td>z-z</td><td>d</td></tr>
</table>

Sway and deflection limit

Besides all the constraints from the Eurocode, there are two more constraints that are the maximum sway of the frame allowed and the maximum deflection of beams. These two constraints are calculated as follows:

$$\text{max sway} = \frac{\text{frame height}}{400}\ [\text{mm}] \tag{30}$$

$$\text{max deflection} = \frac{\text{bay length}}{300}\ [\text{mm}] \tag{31}$$

The maximum sway of the frame and the maximum deflection of beams are shown in Figure 15.

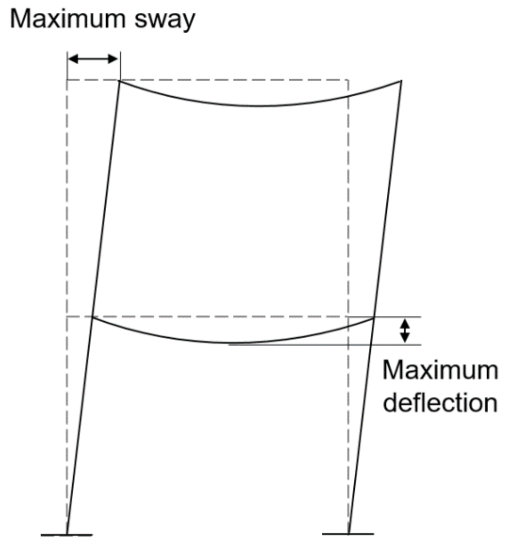

Figure 15. *Maximum sway and maximum deflection*

3.4 Considerations

In the studied steel frames, some considerations are applied to simplify their calculations and solution. First of all, only linear analysis is going to be made, which means that a linear relation holds between applied forces and displacements.

Joints between frame members are rigid and the supports are fixed. The eccentricities of the joints are not taken into account and they are not optimized.

As the main goal of this study is to find the optimum design of the frame to have the minimum weight the distribution of profiles does not need to be symmetric due to the load distributions is not always symmetric.

4. OPTIMIZATION METHODS

Optimization problems can be solved using gradient-based methods that need the function derivatives to find the optimum solution or gradient-free methods that imitate some mechanisms observed in the nature or uses heuristics. In this section, these two methods are explained and compared the advantages and disadvantages.

4.1 Gradient-based methods

Gradient-based methods solve constrained and unconstrained nonlinear optimization problems using the function derivatives. These methods employ numerical techniques to calculate the direction in the design space where to search for a better estimate of the optimum solution. This search of direction relies on the estimation of the value of the gradient of the objective function at a given point.

These methods have the advantage that they are applicable to a broader class of problems than linear programming. However, gradient-based methods are more mathematically sophisticated and difficult than linear programming [29].

The performance of these methods depends on the initial values of the design variables, which need to be feasible. Different initial values can lead to different solutions due to the algorithm can be stuck in a local minimum, for this reason, it is necessary to run several optimization runs with different initial values.

Some examples of these methods are:

- Steepest descent method
- Conjugate gradient method
- Newton´s method
- Quasi-Newton method
- Trust region method

Newton's, quasi-Newton and trust region methods are explained below since the algorithms used in this thesis are based on them.

4.1.1 Newton's method

Newton's method uses second derivatives or the Hessian matrix. This is in contrast to the steepest descent method and conjugate gradient method, which are first-order methods and they require only first derivatives or gradient information. When Newton's method converges, it does it faster than first-order methods.

The idea is to construct a quadratic approximation to the function $f(x)$ and minimize the quadratic. The quadratic approximation is:

$$q(x) = f(x_k) + \nabla f(x_k)^T(x - x_k) + \frac{1}{2}(x - x_k)^T \nabla^2 f(x_k)(x - x_k) \tag{32}$$

Assuming that $\nabla^2 f(x_k)$ is positive definite, the minimum of $q(x)$ is found by setting $\nabla q = 0$ which yields

$$[\nabla^2 f(x_k)]d_k = -\nabla f(x_k) \tag{33}$$

Where $d_k = x_{k+1} - x_k$ and solving the previous equation for d_k the new point is obtained:

$$x_{k+1} = x_k + d_k \tag{34}$$

With this new point, the gradient and Hessian are recalculated to update again the point. However, Newton's method has some disadvantages, it does not guarantee to converge and needs to compute not only the gradient but also the Hessian matrix, which contains $n(n+1)/2$ second order derivatives [23][30].

4.1.2 Quasi-Newton methods

Quasi-Newton methods use a Hessian-like matrix F_k but without calculating second order derivatives instead, they just use gradient information and approximate the Hessian matrix.

This method starts initializing the Hessian to the identity matrix and updating it in each iteration. The basic idea is that in each iteration Hessian information is added to this matrix. Since this method actually needs the inverse of the Hessian, it will use the inverse of F_k for all the calculations [23][30].

4.1.3 Trust region methods

Trust region methods are a different approach to solve the weakness of the pure form of Newton's method. Newton's method may break down when the Hessian is not positive definite or when the function is highly nonlinear because it is trying to minimize $q(x)$ outside the region where the quadratic approximation is valid.

To overcome this problem, the trust region method minimizes $q(x)$ within a region around the current search point x_k where the quadratic approximation for local minimization is "trusted" to be [23][30].

The size of the "trust" region change depending on how well the model agrees with actual function evaluations.

4.2 Gradient-free methods

Gradient-free algorithms try to imitate the mechanism observed in nature or use heuristics. Some gradient-free methods are simply a structured random search that instead of moving from one design point to the next, it makes use of a population of design points [31]

These methods can not guarantee that the solution found is the global optimum but they can come with many good solutions and multiple local optima. They are used due to their ability to solve problems too difficult to solve with gradient-based methods and can handle multi-modal problems or discrete and mixed discrete-continuous design variables.

Some examples of these methods are [32]:

- Evolutionary algorithms
- Harmony search
- Simulated annealing
- Particle swarm optimization
- Simplex gradient methods
- Nelder-Mead simplex

Evolutionary algorithms and particle swarm optimization are explained below since the algorithms used in this thesis are based on them.

4.2.1 Particle swarm optimization

Particle swarm optimization (PSO) is a stochastic, population-based algorithm developed in 1995 by James Kennedy and Russell Eberhart created to optimize continuous nonlinear functions [33][34].

This algorithm is based on social psychological principles and it does not use selection like evolutionary algorithms, all members of the population survive from the beginning of a trial until the end. The quality of the solutions improves over time due to the interaction between population members. However, it has an important relationship with evolutionary algorithms and artificial life in particular, to bird flocking, fish schooling and swarming theory.

This method applies the concept of swarm intelligence to problem-solving. Swarm intelligence “is the property of a system whereby the collective behaviors of (unsophisticated) particles interacting locally with their environment cause coherent functional global patterns to emerge” [31][35].

PSO emulates the social behavior of fishes and birds and initializes a set of candidate solutions for searching the optimum solution. Each particle represents a candidate solution and they are scattered around the search-space moving around trying to find the optimum position [36].

The movements of the particles are affected by their cognitive desire to search individually and the collective action of the group or its neighbors as shown in Figure 16. The inertia makes the particle move in the same direction and with the same velocity, the personal influence makes it move to a previous position better than the actual and the social influence makes it follow the best neighbor’s direction.

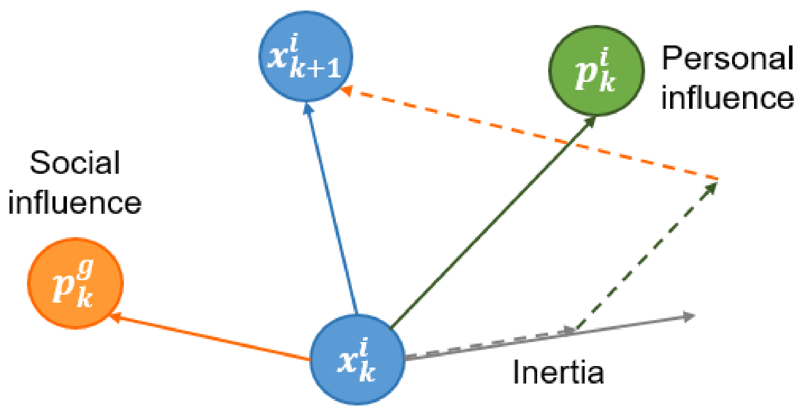

Figure 16. *Particle swarm*

4.2.2 Evolutionary algorithms

Evolutionary algorithms are gradient-free algorithms that consist of several heuristic searches and imitate some aspects of natural evolution. These algorithms follow four general steps: reproduction, recombination, mutation and selection, view Figure 17. They use a fitness function to determine the conditions that support survival [32][37],

The basic idea is that those individuals matching certain criteria will reproduce and after some generations, the population will converge in individuals that best match these criteria.

During the reproduction some imperfections are added, they consist of random mutations in some individuals. Population behavior follows the rules of the Darwin evolution theory.

As this method is based on natural genetics, the terminology used is the same. The individuals in a population are called chromosomes and there are made of units called genes, each of which encodes a particular feature of the organism [38].

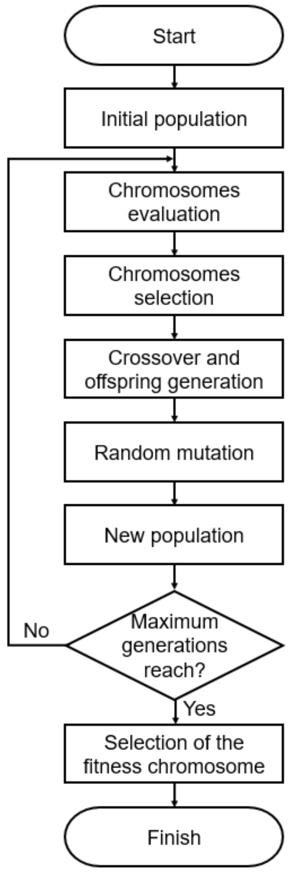

Figure 17. *Evolutionary algorithm*

5. TWO-PHASE APPROACH

The procedure used in this study is the two-phase approach [16][17][18]. It consists of first solving the continuous problem finding solutions not available in the profile catalogs and after this, using this optimum result to solve the discrete problem and find the better option in the available commercially profiles, see Figure 18. This chapter focuses on the algorithms used to solve the minimization problem.

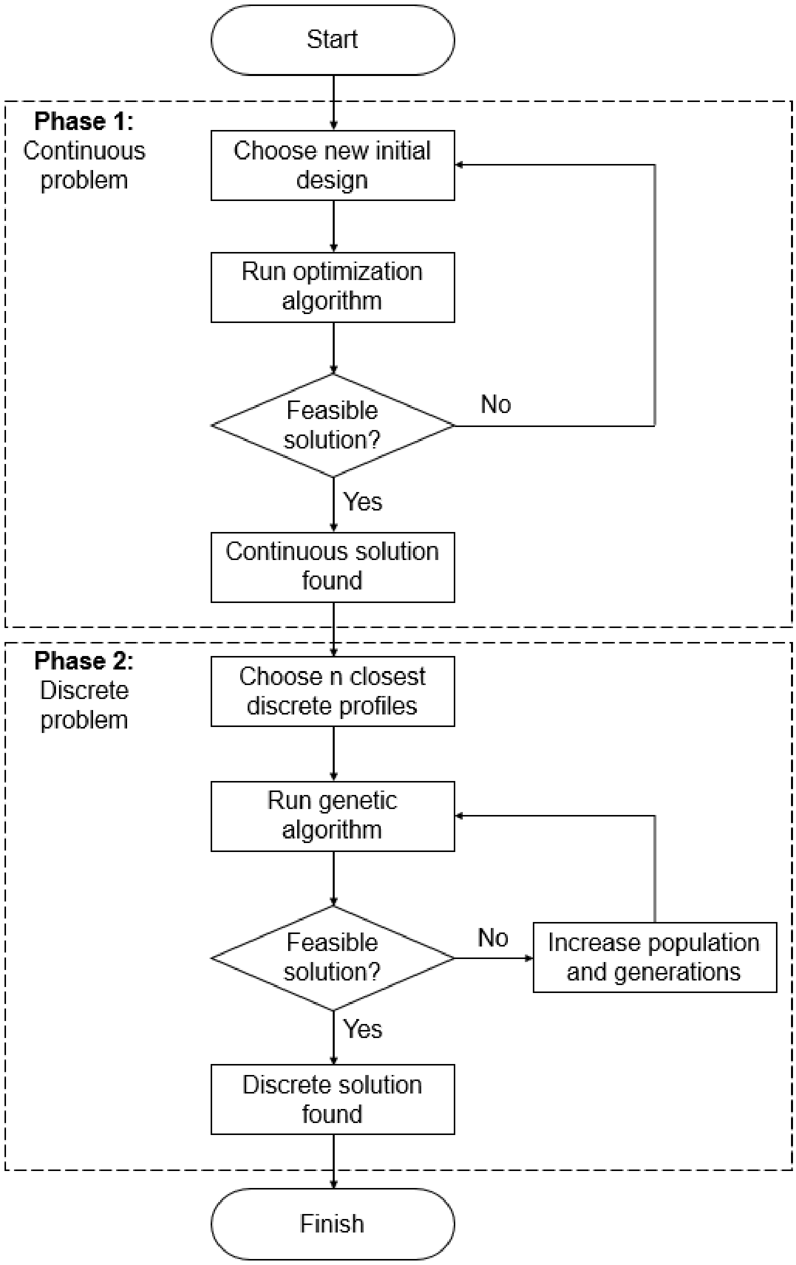

Figure 18. *Flow chart two-phase approach*

5.1 Phase 1: Continuous problem

To solve the continuous problem 3 different algorithms already implemented in Python, are used. All these algorithms can handle constrained minimization problems and how they work is explained below.

5.1.1 SLSQP

Sequential least-squares programming (SLSQP) is an optimizer algorithm written by Dieter Kraft that solves constrained nonlinear minimization problems and is based on the Quasi-Newton method [39].

This optimization algorithm is found in the sciPy library of Python and the parameters required to solve the minimization problem are the objective function to be minimized, initial guess for the design variables, list of inequality constraints, list containing the lower and upper bound of all the design variables and the maximum number of iterations [40].

This algorithm returns the value of the design variables that minimize the objective function, the final value of the objective function and the number of iterations needed to solve the problem.

5.1.2 COBYLA

Constrained optimization by linear approximation (COBYLA) is a trust region algorithm developed by Powell in 1994 [41] and based on linear approximations of the objective function and the constraints. It is found in the sciPy library of Python.

This algorithm needs the objective function, initial guess for the design variables, list of inequality constraints, the maximum number of function evaluations and which is the final accuracy in the optimization [42][43][44].

It returns the value of the design variables that minimize the objective function and as the boundaries of the design variables cannot be fixed with this algorithm, they have to be implemented as constraints.

5.1.3 PSO

Particle swarm optimization (PSO) is an optimizer found in the Pyswarm library of Python. This is a gradient-free algorithm, as explained before, and the required parameters to solve the minimization problem are the objective function to be minimized, list with the lower bounds, list with the upper bounds, list of inequality constraints, number of particles in the swarm, maximum number of iterations, the minimum step size of swarm's best position before the search terminates [45].

This algorithm returns the value of the design variables that minimize the objective function and the final value of the objective function.

5.2 Phase 2: Discrete problem

After finding the continuous solution, the discrete problem is solved. Using the optimum solution found in the previous step, the algorithm searches which one of the solutions available in the market is the closest.

5.2.1 Genetic algorithm

The genetic algorithm developed is based on the method created by Osyczka and Krenich in 1999. This algorithm uses tournament selection to solve nonlinear minimization problems instead of penalty function because Osyczka proved that is the most effective for these type of problems [46].

This selection chose the best solution, in reference to the fitness objective function, for the next generation. For the minimization problem, the solution with the smaller fitness value is selected and kept in an intermediate population [47].

This genetic algorithm has been adapted to solve the case study which is a discrete problem and whose possible solutions have to be chosen from the available profiles in the market. Here it is explained how it works [48], [49].

Initial population

First of all the initial population is created, it has 200 chromosomes, each one has the same amount of genes as members have the studied frame and each gen is the member's cross-section area.

This population is created randomly choosing among the closest profiles to the optimum solution. To know which are the closest profiles, the Euclidean distance between available profiles and the continuous optimum solution is calculated. How to calculate the Euclidean distance is explained in chapter 6.

Then a specific amount of the closest profiles is chosen to create the initial population and to search the discrete optimum solution among them, view Figure 19.

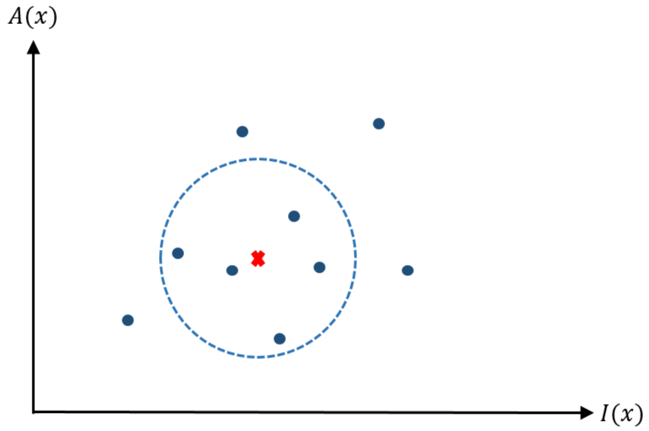

Figure 19. *Discrete solutions closest to the optimum solution*

Tournament selection

Once the initial population is created the tournament selection is done. It consists of some steps that allow the feasible solutions of the initial population to have a greater probability to be chosen for the next generation.

The tournament between two chromosomes is carried out in the following way [46]:

- If both chromosomes are not in the feasible region the one which is closer to the feasible region is taken to the next generation. The values of the objective function are not calculated for either of the chromosomes.
- If one chromosome is in the feasible region and the other one is out of the feasible region the one which is in the feasible region is taken to the next generation. The values of the objective function are not calculated for either chromosomes.
- If both chromosomes are in the feasible region, the values of the objective function are calculated for both chromosomes and the one, which has a better value of the objective function, is taken to the next generation.

The constraint violation function is evaluated as follows:

$$\Psi(x) = \sum_{m=1}^{M} [h_m(x)]^2 + \sum_{k=1}^{K} G_k[g_k(x)]^2 \tag{35}$$

In this thesis, the problem studied has only inequality constraints, which means that the first term of the violation function equals to zero.

$$\Psi(x) = \sum_{k=1}^{K} G_k[g_k(x)]^2 \tag{36}$$

Where: $G_k = 0$ for $g_k(x) \geq 0$ (constraints satisfied) and $G_k = 1$ for $g_k(x) < 0$ (constraints not satisfied).

For the solutions that are in the feasible region, the value of $\Psi(x) = 0$ and for those that are out of the feasible region, the value of $\Psi(x)$ indicates how far the solutions are from the feasible region. In the tournament selection method, a comparison between violated solutions is made and the one that is less violated, so closer to the feasible region, will be chosen for the next generation.

After the tournament selection, the operations of the evolutionary algorithms are made with the new population obtained for the next generation, first the crossover between parents and after it, the mutation [50].

Crossover

It is the main genetic operator. It operates on two chromosomes at a time and generates offspring by combining some features of both chromosomes. The two chromosomes to be combined are called parents and are chosen randomly which means that one parent can be picked more than once to create offspring in the same generation [46].

In the single point crossover, choose a random cutoff point for both parents to split the gens that will be combined to create the offspring. The first offspring keeps the left gens of the parent 1 and the right gens of parent 2. The second offspring keeps the rest of the gens, view Figure 20.

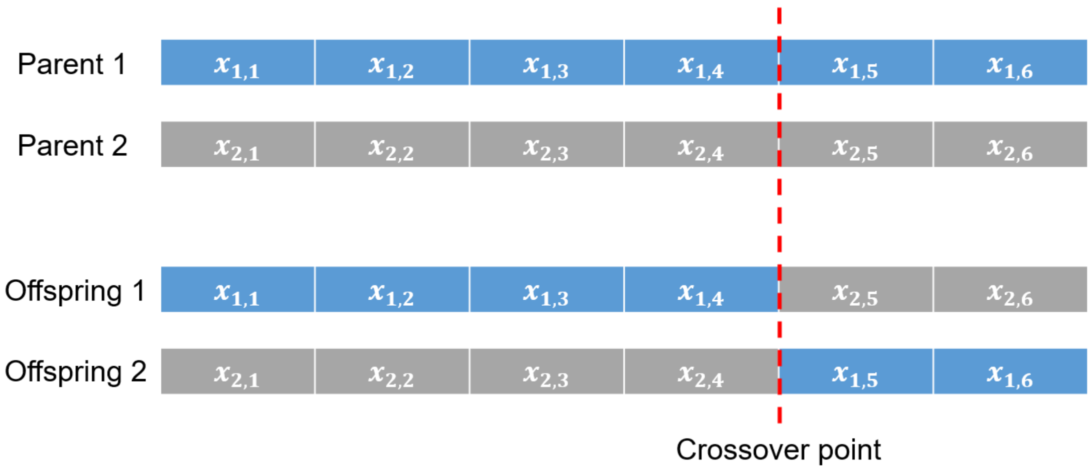

Figure 20. *Genetic algorithm crossover between parents*

Mutation

The mutation is a genetic operator, which produces spontaneous random changes in various chromosomes introducing some extra variability into the population in order to avoid local minimums [46].

Let p_m be the mutation rate, which controls the number of mutated bits in the population. If $p_m = 0.01$, 1% of the bits in the population will undergo mutation. For example, if the population were of 400 chromosomes with 6 bits each 24 bits would mutate in each generation, view Figure 21.

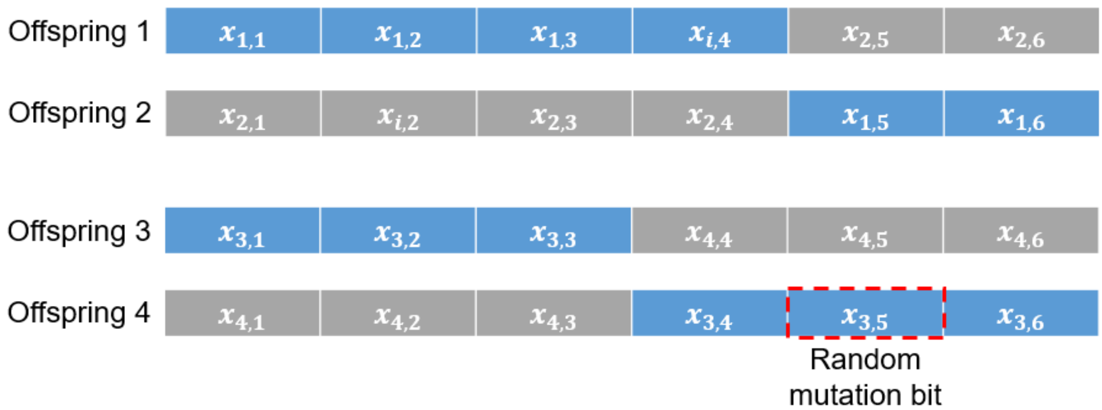

Figure 21. *Random mutations*

To make this mutation in the Python code, a random number between 0 and 1 is assigned to each gen of all chromosomes in the population for the next generation. If this random number is < 0.01 the gen will mutate and take another value of the cross-section areas of the profiles available for this specific member.

Solution

After all generations, the population obtained is evaluated to see which chromosome has the minimum objective function result and the best result that also satisfies all the constraints is chosen as a solution.

6. PROCEDURES

Two different formulations have been written following the same structure but with a big difference, the first one has only one design variable (h) and the second has 5 (h, b, t_f, t_w, r). As the first formulation has only one design variable, to be able to approximate all the other cross-section parameters it is needed to know which kind of profile family is it.

Both formulations find a continuous and discrete solution using the different algorithms explained in the previous section to be able to compare which one works better.

6.1 First formulation

This formulation is faster since there is only one design variable to optimize per member. For this reason, it is needed to decide first which kind of profile is going to be each member.

The beams are fixed as IPE profiles and the columns as HEA because these are the profiles commonly use as beams and columns.

6.1.1 Continuous part

The continuous problem is solved with the three different algorithms explained in section 5.1, SLSQP, COBYLA and PSO.

The initial profiles are HEA 400 for columns and IPE 400 for beams. The design variables are stored in X, a vector containing the h variable of each frame member.

$$X = [h_1, h_2 \dots h_n] \tag{37}$$

In each optimization iteration, these design variables are changed, all the structural calculations are made again and all the constraints checked. This process finishes when the optimum solution is found.

The bounds of each frame member are different depending on which type of member is. Different bounds are assigned to beams or columns depending on which profile family they are.

All the bounds of each member are stored in a vector called bounds. In addition, the lower and upper bounds are stored in separated vectors because some algorithms required just a vector of bounds and others the lower and upper bounds separately.

The h bounds for IPE profiles are [80, 600] and for HEA [96, 990].

All the constraints are stored in a vector call cons and all of them have to be greater than or equal to 0 and these constraints are:

- Frame sway: calculates the maximum sway obtained on the frame.

$$\text{constraint 1} = \frac{\text{frame height}}{400} - \text{frame sway} \geq 0 \tag{38}$$

- Beam deflection: calculates the maximum vertical deflection of each beam.

$$\text{constraint 2} = \frac{\text{bay length}}{300} \text{ - beam deflection} \geq 0 \tag{39}$$

- Eurocode 3 constraints: the utilization ratio for the cross-section, buckling and lateral buckling strength of each member are calculated in the program following the formulas explain in section 3.3 and stored in an internal vector call r. All these ratios have to be less than or equal to 1 but to add them in the constraints vector, the following modification needs to be done:

$$\text{constraint 3} = [1 - r_1, 1 - r_2, 1 - r_3] \geq 0 \tag{40}$$

  Where:

$$r_1 = \max\left(\frac{N_{Ed}}{N_{c,Rd}}, \frac{V_{Ed}}{V_{c,Rd}}, \frac{M_{Ed}}{M_{c,Rd}}\right) \leq 1 \tag{41}$$

$$r_2 = \frac{N_{Ed}}{N_{b,Rd}} \leq 1 \tag{42}$$

$$r_3 = \frac{M_{Ed}}{M_{b,Rd}} \leq 1 \tag{43}$$

- IPE and HEA boundaries: these boundaries need to be implemented as constraints only for COBYLA algorithm.

6.1.2 Discrete part

To solve the discrete part the 5 closest profiles to the continuous optimum solution are selected to use them in the genetic algorithm. It has been decided to be 5 because it was checked with 10 and 20 profiles but the program required more time and the results obtained were the same.

To know which are the closest profiles, the Euclidean distance between the profiles and the optimum solution is calculated with the following formula:

$$d_j = \sqrt{\left(\frac{A_j - A_{opt}}{A_{opt}}\right)^2 + \left(\frac{I_j - I_{opt}}{I_{opt}}\right)^2} \tag{44}$$

The Euclidean distance is calculated with the cross-section area and the second moment of area because the area is the parameter that has to be minimized and the constraints depend also on the second moment of area.

The genetic algorithm is run during 400 generations and with a population of 200 chromosomes, which means 80000 function evaluations to ensure that an optimum solution is found.

6.2 Second formulation

Like the first formulation, the algorithms used to solve the continuous part are SLSQP, COBYLA and PSO and for the discrete, a genetic algorithm.

6.2.1 Continuous part

The initial profiles are HEA 400 for columns and IPE 400 for beams. The design variables are stored in X, a list containing the 5 variables of each frame member.

$$X = [h_1, b_1, t_{f,1}, t_{w,1}, r_1 \dots h_n, b_n, t_{f,n}, t_{w,n}, r_n] \tag{45}$$

In this formulation, each profile of the frame can be from a different type of family and as the correlation between parameters is different for each family, the bounds are different too. To cope with this difference, the lower and upper bounds established are the least restrictive and to ensure that all the parameters are inside the boundaries some outer approximations have been established.

The lower and upper bounds for the design variables h, b, t_f, t_w, r are:

- lower bound = [80, 46, 5.2, 3.8, 5]
- upper bound = [1008, 310, 40, 21, 30]

The outer approximations used to ensure that the results obtained are inside feasible boundaries had been done using the correlation between parameters shown in Figure 22, Figure 23, Figure 24 and these feasible areas are implemented as linear constraints.

These outer approximations have been found visually, using the upper and lower limits of the design variables and the linear correlations of the profile families closer to these outer approximations.

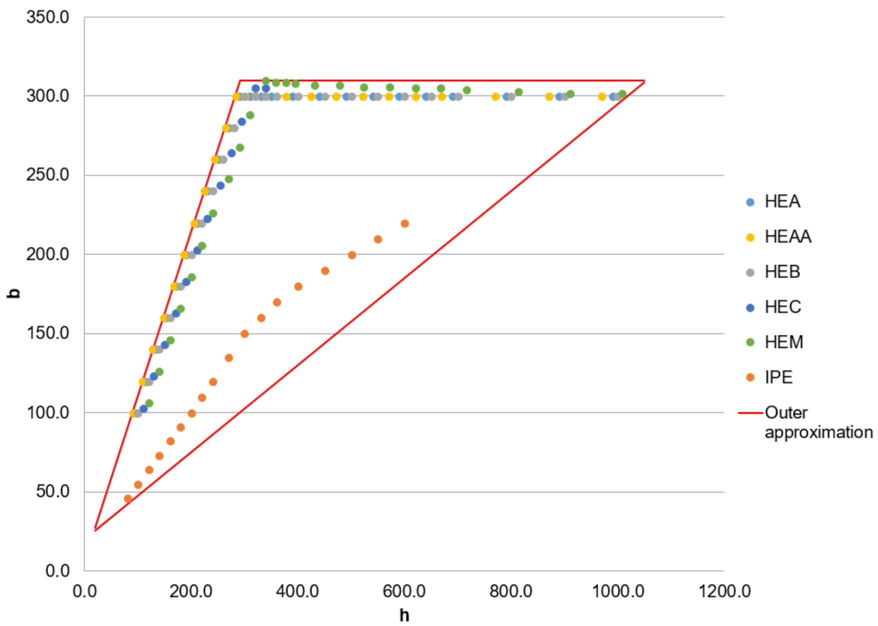

Figure 22. *Outer approximation between h-b variables*

The outer approximation between h and b is defined with the following constraints as well as with the corresponding lower and upper bounds:

$$\text{constraint 4} = -b + 1.0403\,h + 6.3 \leq 0 \quad (46)$$

$$\text{constraint 5} = b - 0.275\,h - 20 \leq 0 \quad (47)$$

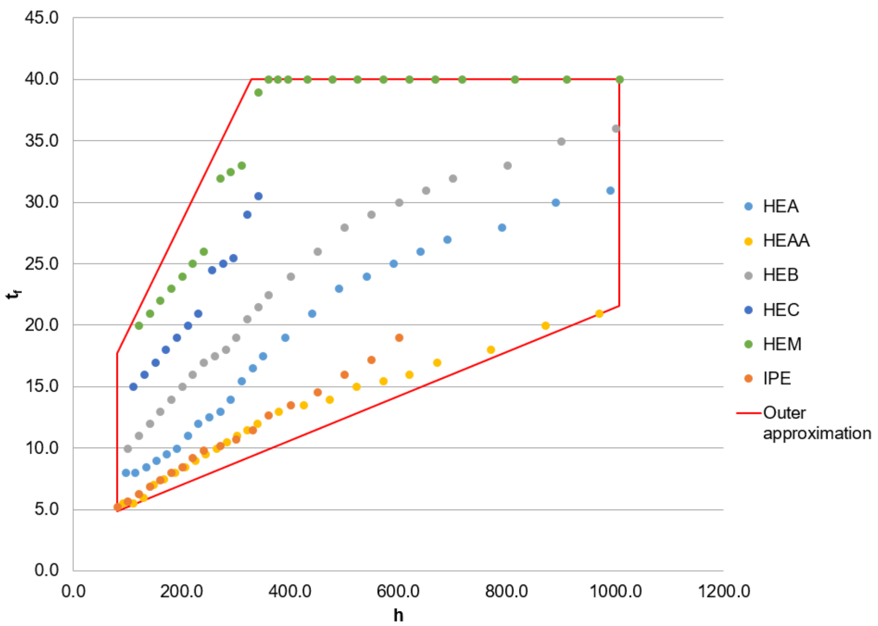

Figure 23. *Outer approximation between h-t_f variables*

The outer approximation between h and t_f is defined with the following constraints as well as with the corresponding lower and upper bounds:

$$\text{constraint 6} = -t_f + 0.09\, h + 10.5 \leq 0 \tag{48}$$

$$\text{constraint 7} = t_f - 0.018\, h - 3.4 \leq 0 \tag{49}$$

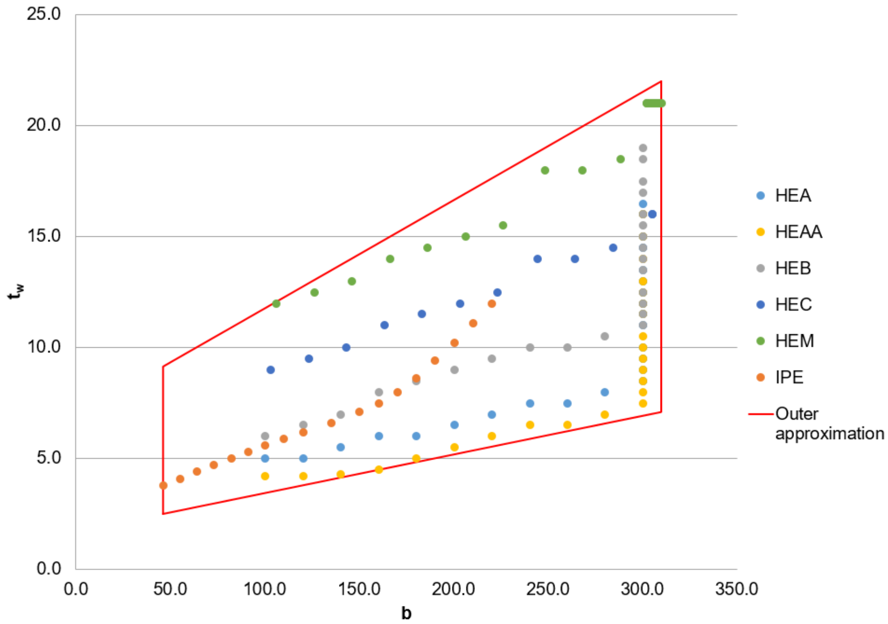

Figure 24. *Outer approximation between b-t_w variables*

The outer approximation between b and t_w is defined with the following constraints as well as with the corresponding lower and upper bounds:

$$\text{constraint 8} = -t_w + 0.0487\, b + 6.9 \leq 0 \tag{50}$$

$$\text{constraint 9} = t_w - 0.0174\, b - 1.7 \leq 0 \tag{51}$$

All the other constraints are the same implemented in the first formulation and explained in chapter 6.1.1

6.2.2 Discrete part

To solve the discrete part the 30 closest profiles to the continuous optimum solution are selected to use them in the genetic algorithm. It has been tried selecting 20 profiles but sometimes the discrete optimum solution that satisfied all the constraints was among them.

To know which are the closest profiles, the Euclidean distance between the profiles and the optimum solution is calculated with the following formula:

$$d_j = \sqrt{\left(\frac{h_j - h_{opt}}{h_{opt}}\right)^2 + \left(\frac{b_j - b_{opt}}{b_{opt}}\right)^2 + \left(\frac{t_{f,j} - t_{f,opt}}{t_{f,opt}}\right)^2 + \left(\frac{t_{w,j} - t_{w,opt}}{t_{w,opt}}\right)^2} \quad (52)$$

The r parameter is not considered in this formula because it does not have a significant influence.

In this second formulation, the Euclidean distance is calculated with the design variables because the continuous solution has all the optimum design variables but they can be very far of any available profile in the market. For this reason, it has been decided to select the 30 closest profiles for the genetic algorithm, this way there are profiles from all the families and it ensures that there will be some profiles that can satisfy all the constraints.

For this formulation, the genetic algorithm is also run during 400 generations and with a population of 200 chromosomes.

7. CASE STUDIES

The problem to solve is a steel structure with a variable number of bays, storeys and different loads applied. To check the program three different case studies had been designed, starting with a small frame and increasing the complexity until the last one that pretends to be a representation of a feasible building.

7.1 Small Frame

The first problem is a small frame of 1 bay, 2 storeys, 4 m bay length and 3.5 m storey height. It has a uniform load in both beams of 200 kN/m, view Figure 25.

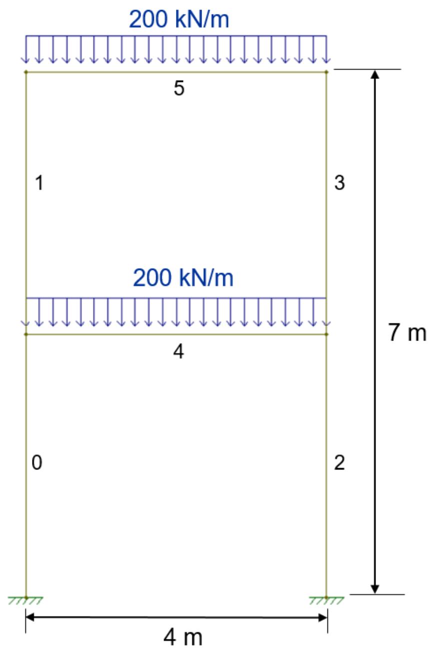

Figure 25. *Small frame*

7.2 Medium frame

The second frame to study is the same one studied by van Mellaert et al [51] and which optimum weight was found to be 6132 kg, but in that case, all profiles were HEA and as in this study they can be from different families.

Another difference with the frame studied by van Mellaert et al is that the profiles distribution was enforced to be symmetric, for these reasons it was expected to find better results with the program developed in this thesis.

The frame has 3 bays of 6 m length and 3 storeys of 3.5 m height. It has a uniform load of 50.1 kN/m on all beams and punctual loads 22.05 kN on the left and right nodes, view Figure 26.

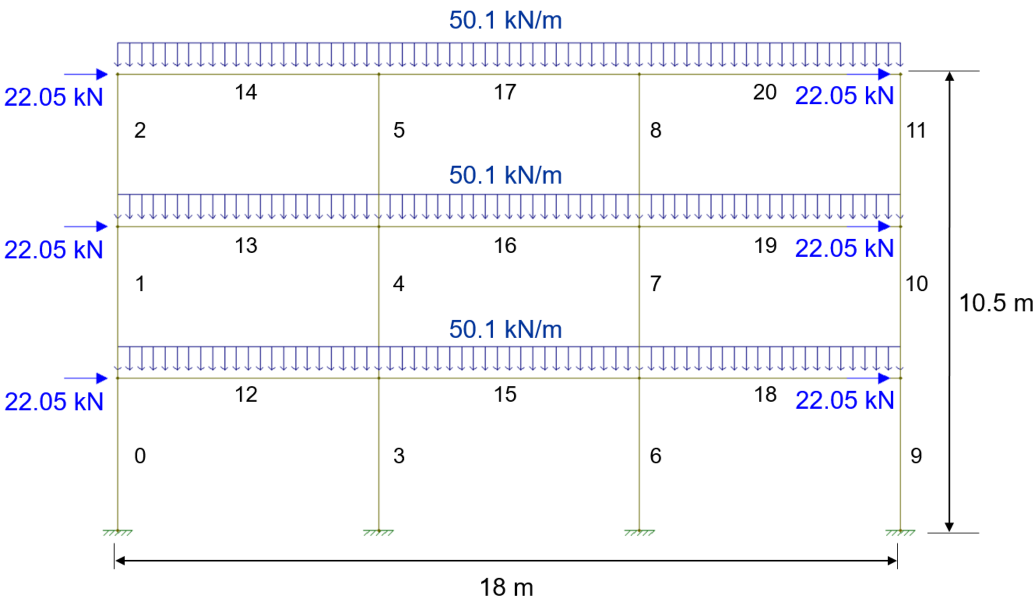

Figure 26. *Medium frame*

7.3 Large frame

The last frame to study simulates a possible building, it has 4 bays of 6 m length and 8 storeys of 3.5 m height.

The loads applied have been calculated using Limit State Design, where there is structural weight (G) as a permanent load and as changeable loads, snow (Q_1) for the last storey and live weight (Q_2) on the others.

The distance between frames is 10 m, which means that the linear loads are: $G =$ 50 kN/m, $Q1 =$ 20 kN/m, $Q2 =$ 25 kN/m.

The coefficient for a changeable load is $\gamma_Q = 1.5$ and for the permanent $\gamma_G = 1.35$.

The rooftop has only structural weight and snow and the load is calculated as follows:

$$\gamma_G \cdot \mathrm{G} + \gamma_Q \cdot Q_1 = 1.35 \cdot\ 50 kN/m + 1.5 \cdot 20\ kN/m = 97.5\ kN/m \tag{53}$$

All the other beams have structural weight and live weight and the load is calculated as follows:

$$\gamma_G \cdot \mathrm{G} + \gamma_Q \cdot Q_2 = 1.35 \cdot\ 50 kN/m + 1.5 \cdot 25\ kN/m = 105\ kN/m \tag{54}$$

The wind is also considered as a uniform load but applied on the left columns and it is 6 kN/m. Figure 27 shows the frame to study.

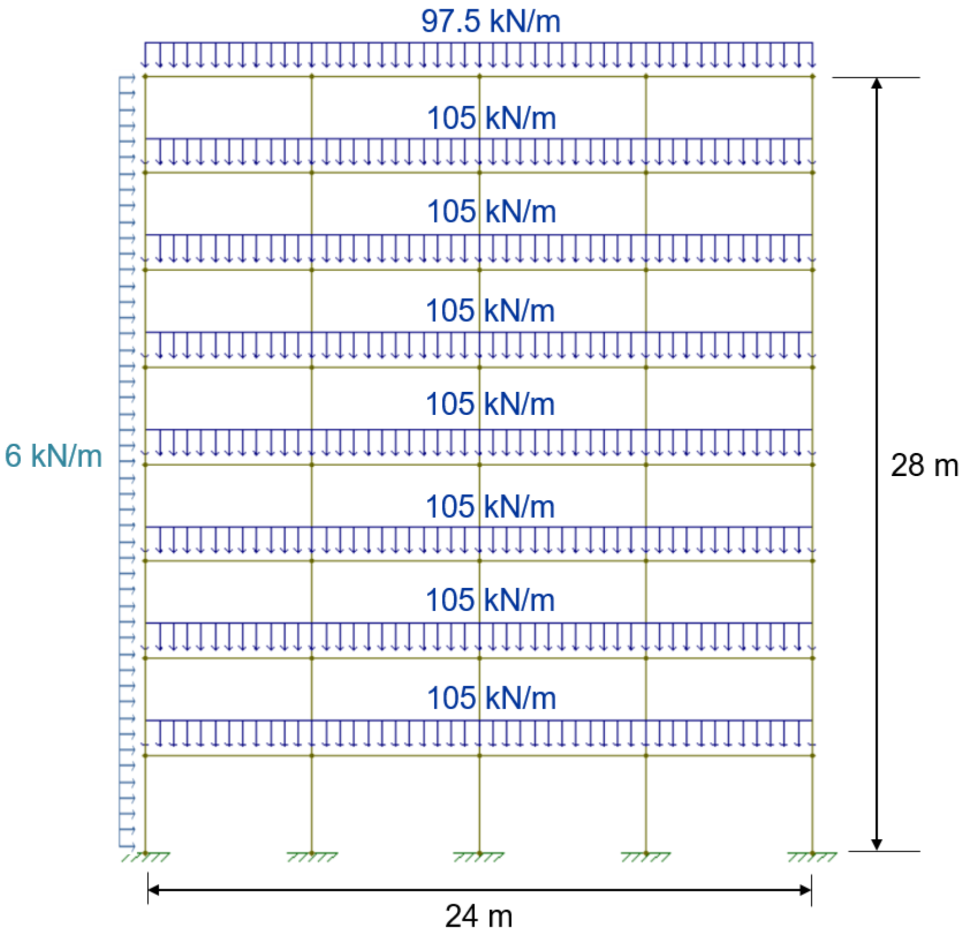

Figure 27. *Large frame*

8. RESULTS

The three different case studies designed were solved with the two different formulations developed to be able to compare them and find out which one was better.

8.1 First formulation

In the continuous part of this formulation, the columns are HEA profiles but the number of profile indicates the h value and this is the value needed to calculate all the other cross-section parameters with the formulas indicated in chapter 3.2.

8.1.1 Small frame

Continuous

For the small frame, three different algorithms have been run 10 times each one, each time the initial point for the optimization is different because there is a random function that sets it inside the design variables boundaries. For SLSQP and PSO the results obtained are always the same, but COBYLA shows different results each time, so there are local minimums. Table 6 show the best results obtained for each optimization algorithm.

Table 6. *Continuous profiles for the small frame*

Member	SLSQP	COBYLA	PSO
0	HE 159.37 A	HE 163.32 A	HE 159.37 A
1	HE 123.60 A	HE 123.57 A	HE 123.60 A
2	HE 159.37 A	HE 414.76 A	HE 159.37 A
3	HE 123.60 A	HE 123.63 A	HE 123.60 A
4	IPE 429.80	IPE 395.63	IPE 429.80
5	IPE 443.29	IPE 443.78	IPE 443.29
Frame weight	1002.12 kg	1263.63 kg	1002.12 kg

Table 7 show the mean, standard deviation, minimum and maximum values of the computation time needed to run each algorithm and it proves that SLSQP is better because it finds the solution 5 times faster than COBYLA and 45 times faster than PSO. It is also important to see that PSO has a really high standard deviation which means that the computation time can be quite different each time.

Table 7. *Computation time small frame*

	SLSQP	COBYLA	PSO
Mean	4.707 s	20.488 s	215.490 s
Standard deviation	0.426	0.278	34.842 s
Minimum time	3.930 s	20.175 s	166.553 s
Maximum time	5.355 s	21.077 s	279.283 s

Discrete

Each time that a continuous solution has been found the genetic algorithm has been run after it to find the discrete solution. However, the results for the discrete part only include the ones obtained after the SLSQP and PSO algorithms, as they are able to reach the same solution each time and do not get stuck with the local like COBYLA.

Table 8 shows the best result obtained with the genetic algorithm after running 10 times all the problems with the SLSQP and 10 times with the PSO algorithms to solve the continuous part.

The Euclidean distance between the optimum continuous profiles and the discrete are also shown in this table and they are very low which means that are close to the optimum continuous solution.

For example beam 5 is the member with a lower Euclidean distance and it is because the IPE 450 has $h = 450$ mm and the continuous solution was $h = 443.29$ mm. It happens the same for the columns, column 0 is a HE 180 A profile with $h = 171$ mm and the h obtained in the continuous solution is 159.37 mm and these two values are also very close.

Table 8. *Discrete profiles for the small frame*

Member	Discrete solution	Euclidean distance
0	HE 180 A	0.157
1	HE 140 A	0.162
2	HE 180 A	0.157
3	HE 140 A	0.162
4	IPE 450	0.256
5	IPE 450	0.114
Frame weight	1041.89 kg	-

Table 9 shows the mean, standard deviation, minimum and maximum values of the computation time needed to run the genetic algorithm these 20 times.

When comparing the mean of computation time obtained in the discrete part with the time for the SLSQP algorithm of the continuous part that was 4.707 s, it is observed a high increase.

This is because the continuous algorithm stops when it finds the optimum solution but the genetic algorithm needs to be run a fix amount of times, in this case 400 times with a population of 200, to improve the probability to find the optimum solution when reaching the last generation.

Table 9. *Computation time small frame*

	Discrete solution
Mean	1654.955 s
Standard deviation	23.763 s
Minimum time	1621.288 s
Maximum time	1695.109 s

8.1.2 Medium frame

Continuous

For the medium frame in the continuous problem, the PSO algorithm is not able to find a feasible solution even increasing the number of maximum iterations. For this reason, only SLSQP and COBYLA are used in this study case and also each algorithm has been run 10 times.

As the frame is more complex, the SLSQP gets stuck with local minima, however, the results obtained are significantly better than with COBYLA.

Table 10 shows the best results for each algorithm and the frame weight obtained with COBYLA is a 56% higher than with SLSQP. This significant difference between the two algorithms is due to the disparity of profiles obtained for some of the members.

Although the majority of the beams have almost the same profile, it does not happen the same with the columns. Except for column 2 that has the same profile with both algorithms and column 6 that has a lighter profile with COBYLA, all the others have much bigger profiles with COBYLA than with SLSQP and therefore this implies an increase of the frame weight.

Table 10. *Continuous profiles for the medium frame*

Member	SLSQP	COBYLA
0	HE 118.41 A	HE 183.25 A
1	HE 103.05 A	HE 399.86 A
2	HE 96.00 A	HE 96.0 A
3	HE 172.96 A	HE 563.34 A
4	HE 317.88 A	HE 603.58 A
5	HE 116.02 A	HE 487.73 A
6	HE 541.13 A	HE 167.03 A
7	HE 143.74 A	HE 588.71 A
8	HE 112.56 A	HE 443.41 A
9	HE 127.78 A	HE 803.66 A
10	HE 270.00 A	HE 552.65 A
11	HE 260.51 A	HE 416.63 A
12	IPE 426.08	IPE 413.14
13	IPE 428.05	IPE 409.31
14	IPE 409.91	IPE 425.27
15	IPE 430.20	IPE 406.50
16	IPE 387.38	IPE 402.58
17	IPE 407.77	IPE 392.36
18	IPE 371.49	IPE 419.18
19	IPE 387.04	IPE 402.48
20	IPE 391.63	IPE 391.49
Frame weight	5545.08 kg	8666.26 kg

Table 11 shows the mean, standard deviation, minimum and maximum values of the computation time needed to run each algorithm. SLSQP needs 2.5 more computation time than COBYLA and it has a higher standard deviation but finds better results.

Table 11. *Computation time medium frame*

	SLSQP	COBYLA
Mean	196.693 s	76.737 s
Standard deviation	61.156 s	0.719 s
Minimum time	133.105 s	76.291 s
Maximum time	325.733 s	78.832 s

Discrete

For the discrete part, only the results obtained after running the SLSQP are include because the continuous solution with COBYLA was worst which means that the profiles selected for the discrete would not be the closest to the optimum solution, Table 12 shows the best result obtained with the genetic algorithm.

The Euclidean distance in this case is also very close to the continuous optimum solution, the member that is farthest away is the beam 17 that has IPE 450 in the discrete solution and IPE 407.77 for the continuous solution. However, both heights are quite close and IPE 450 is the first discrete profile with a height bigger than the continuous solution.

Table 12. *Discrete profiles for the medium frame*

Member	Discrete solution	Euclidean distance
0	HE 140 A	0.349
1	HE 120 A	0.281
2	HE 100 A	0.090
3	HE 200 A	0.276
4	HE 300 A	0.243
5	HE 140 A	0.451
6	HE 600 A	0.395
7	HE 160 A	0.110
8	HE 120 A	0.119
9	HE 140 A	0.074
10	HE 280 A	0.061
11	HE 280 A	0.090
12	IPE 450	0.299
13	IPE 450	0.276
14	IPE 450	0.505
15	IPE 450	0.252
16	IPE 400	0.263
17	IPE 450	0.534
18	IPE 400	0.478
19	IPE 400	0.267
20	IPE 400	0.212
Frame weight	6095.30 kg	-

Table 13 shows the mean, standard deviation, minimum and maximum values of the computation time needed to run the genetic algorithm after running 10 times the SLSQP.

Table 13. *Compution time medium frame*

	Discrete solution
Mean	6162.094 s
Standard deviation	44.753 s
Minimum time	6129.698 s
Maximum time	6162.166 s

8.1.3 Large frame

For the large frame, any of the continuous algorithms is able to find a feasible solution for this reason, only the discrete part had been solved.

To find the best solution the genetic algorithm had been run searching among all the available profiles as there was no continuous solution to use for calculating the Euclidean distance needed to select the 5 closest discrete profiles.

Discrete

The genetic algorithm has been run 3 times and the best result for the discrete part is shown in Table 14. As there are more members, it had been decided to use a population of 300 chromosomes and 600 generations.

The mutation ratio was needed to be lowered to 0.5 % because this frame has 72 gens per chromosome and with a 1% mutation ratio it was impossible to find a feasible solution as too many chromosomes were having a mutated gene.

When lowering the mutation ratio, it allowed having less chromosomes randomly mutating and more feasible solutions were staying for the next generation.

Table 14. *Discrete profiles for the big frame*

Member	Discrete solution	Member	Discrete solution
0	HE 320 A	36	HE 320 A
1	HE 320 A	37	HE 550 A
2	HE 320 A	38	HE 340 A
3	HE 320 A	39	HE 450 A
4	HE 320 A	40	IPE 500
5	HE 320 A	41	IPE 550
6	HE 320 A	42	IPE 500
7	HE 320 A	43	IPE 550
8	HE 550 A	44	IPE 500
9	HE 450 A	45	IPE 500
10	HE 400 A	46	IPE 500
11	HE 320 A	47	IPE 500
12	HE 320 A	48	IPE 600
13	HE 320 A	49	IPE 500
14	HE 320 A	50	IPE 500
15	HE 320 A	51	IPE 500
16	HE 600 A	52	IPE 500
17	HE 450 A	53	IPE 500
18	HE 400 A	54	IPE 500
19	HE 320 A	55	IPE 500
20	HE 320 A	56	IPE 500
21	HE 320 A	57	IPE 500
22	HE 320 A	58	IPE 500
23	HE 320 A	59	IPE 500
24	HE 500 A	60	IPE 500
25	HE 450 A	61	IPE 500
26	HE 650 A	62	IPE 500
27	HE 320 A	63	IPE 500
28	HE 450 A	64	IPE 500
29	HE 320 A	65	IPE 550
30	HE 340 A	66	IPE 550
31	HE 320 A	67	IPE 600
32	HE 360 A	68	IPE 500
33	HE 500 A	69	IPE 550
34	HE 320 A	70	IPE 500
35	HE 600 A	71	IPE 500
Frame weight	34698.91 kg	-	-

Table 15 shows the mean, standard deviation, minimum and maximum values of the computation time needed to run the genetic algorithm 3 times.

Table 15. *Computation time large frame*

	Discrete solution
Mean	37476.553 s
Standard deviation	579.152 s
Minimum time	36824.94697 s
Maximum time	38232.10374 s

8.2 Second formulation

In this second formulation, the design variables are the 5 cross-section parameters which means that in the continuous part the results obtained are the optimum values of these variables but they can be quite far from any commercial section.

8.2.1 Small frame

Continuous

For the small frame, the three different algorithms used for solving the continuous part have been run 10 times each and the best results obtained are included in Table 16. The best solution is found with the SLSQP and after it with COBYLA.

In contrast to the results obtained with the first formulation, with the second formulation PSO is not able to reach the same results as SLSQP furthermore the results obtain are worse than with COBYLA. This different response of the PSO algorithm can be because in the second formulation there are 5 design variables per member instead of 1 so this algorithm works better with less variables.

Table 16. *Continuous profiles for the small frame*

	SLSQP					COBYLA					PSO				
Member	**h**	**b**	**t_f**	**t_w**	**r**	**h**	**b**	**t_f**	**t_w**	**r**	**h**	**b**	**t_f**	**t_w**	**r**
0	182.97	196.65	6.69	5.12	5.00	216.09	185.84	7.29	4.93	9.71	178.78	189.18	21.95	9.34	23.08
1	139.57	151.49	5.91	4.34	5.00	343.35	143.38	9.58	4.19	5.00	175.66	128.89	9.74	10.43	22.59
2	182.97	196.65	6.69	5.12	5.00	358.64	161.50	9.86	4.51	5.00	180.61	147.85	11.80	13.22	13.62
3	139.57	151.49	5.91	4.34	5.00	270.43	164.02	9.19	4.55	5.00	502.94	167.02	18.62	4.73	22.79
4	476.16	203.34	11.97	5.24	5.00	370.97	183.96	10.64	4.90	7.53	496.32	159.02	14.06	4.98	7.40
5	536.95	190.82	13.07	5.02	5.00	520.44	168.32	12.77	4.63	5.00	574.80	210.83	14.45	8.14	20.03
Frame weight	789.29 kg					848.20 kg					1358.85 kg				

Table 17 shows the mean, standard deviation, minimum and maximum values of the computation time need to run each algorithm. COBYLA is faster and has a smaller standard deviation but is better to use SLSQP and be able to reach a better solution.

Table 17. *Computation time small frame*

	SLSQP	COBYLA	PSO
Mean	82.957 s	22.091 s	988.285 s
Standard deviation	52.854 s	0.904 s	323.615 s
Minimum time	46.337 s	20.412 s	490.838 s
Maximum time	212.155 s	22.849 s	1506.846 s

Discrete

The results of the discrete algorithm showed in Table 18 are the ones obtained after running 10 times the SLSQP and this solution is close to the continuous because the Euclidean distances are very small. For example member 0 in the continuous solution had $h = 182.97$ mm which is close to the discrete HE 180 A profile that has $h = 171$ mm and all the other cross-section parameters are similar too.

Table 18. *Discrete profiles for the small frame*

Member	Discrete solution	Euclidean distance
0	HE 180 A	0.304
1	HE 160 AA	0.532
2	HE 180 A	0.304
3	HE 160 AA	0.532
4	IPE 450	0.378
5	IPE 450	0.342
Frame weight	1036.09 kg	-

Table 19 shows the mean, standard deviation, minimum and maximum values of the computation time needed to solve de discrete part.

Table 19. *Computation time small frame*

	Discrete solution
Mean	1621.820 s
Standard deviation	21.497 s
Minimum time	1590.214 s
Maximum time	1656.572 s

8.2.2 Medium frame

Continuous

While solving the continuous part of the medium frame, it happens the same as in the first formulation, the PSO algorithm is not able to find a feasible solution.

Table 20 shows the best results obtained with the SLSQP and COBYLA. The solution found with COBYLA is almost a 70% heavier than with SLSQP.

Table 20. *Continuous profiles for the medium frame*

	SLSQP					COBYLA				
Member	**h**	**b**	**t_f**	**t_w**	**r**	**h**	**b**	**t_f**	**t_w**	**r**
0	133.63	145.32	5.81	4.23	5.00	812.14	243.34	18.02	5.93	5.00
1	119.54	130.66	5.55	3.97	5.00	319.98	235.49	9.16	5.80	5.00
2	90.35	100.29	5.20	3.80	5.00	884.44	263.22	19.32	6.28	5.00
3	208.51	223.21	7.15	5.58	5.00	607.80	227.43	14.34	5.66	5.00
4	439.57	140.88	11.31	4.15	5.00	360.38	151.23	9.89	4.33	5.00
5	130.66	142.23	5.75	4.17	5.00	448.83	283.32	11.48	6.63	5.00
6	669.85	204.21	15.46	5.25	5.00	493.84	262.69	12.29	6.27	6.20
7	163.43	176.32	6.34	4.77	5.00	826.37	247.25	18.27	6.00	5.00
8	126.37	137.76	5.67	4.10	5.00	872.85	260.03	19.11	6.22	5.00
9	144.85	156.99	6.01	4.43	5.00	180.96	191.59	6.66	5.03	5.00
10	410.48	132.88	10.79	4.01	5.00	627.69	205.23	14.70	5.27	5.00
11	384.24	125.67	10.32	3.89	5.00	398.38	249.15	10.57	6.04	8.70
12	407.62	219.67	10.74	5.52	5.00	585.53	181.02	13.94	4.85	5.00
13	534.11	179.83	13.01	4.83	5.00	269.06	279.21	8.43	6.56	7.48
14	317.48	249.09	9.11	6.03	5.00	227.57	111.13	28.14	3.80	5.00
15	403.27	223.62	10.66	5.59	5.00	563.06	174.84	13.54	4.74	5.00
16	281.50	245.43	8.47	5.97	5.00	193.66	171.16	20.59	4.68	7.54
17	270.38	283.87	8.27	6.64	5.00	662.17	236.88	15.32	5.82	5.00
18	293.61	224.74	8.69	5.61	5.00	204.28	215.00	14.74	5.44	5.00
19	275.43	249.48	8.36	6.04	5.00	620.97	190.77	14.58	5.02	5.00
20	280.41	254.63	8.45	6.13	5.00	524.88	164.34	12.85	4.56	5.00
Frame weight	3838.01 kg					6497.96 kg				

Table 21 shows the mean, standard deviation, minimum and maximum values of the computation time need to run each algorithm.

The computation time difference between the two algorithms is bigger than in the small frame but is still better to use the SLSQP and find a better solution even though it requires more time.

Table 21. *Computation time medium frame*

	SLSQP	COBYLA
Mean	2040.084 s	88.233 s
Standard deviation	436.927 s	1.917 s
Minimum time	1465.210 s	83.522 s
Maximum time	2812.661 s	90.257 s

Discrete

The results for the discrete part are shown in Table 22. Almost all the discrete profiles are close to the continuous solution as the Euclidean distance shows except column 3 and 7 that have bigger Euclidean distances.

Column 3 is an IPE 550 which means $h = 550$ mm while for the continuous solution it was found $h = 208.51$ mm and this big difference made the Euclidean distance to increase.

Table 22. *Discrete profiles for the medium frame*

Member	Discrete solution	Euclidean distance
0	HE 160 AA	0.776
1	HE 120 A	0.370
2	HE 100 AA	0.193
3	IPE 550	17.776
4	IPE 330	0.415
5	HE 180 AA	1.920
6	HE 200 A	1.052
7	IPE 300	4.540
8	HE 160 AA	1.144
9	HE 160 AA	0.350
10	IPE 360	0.667
11	IPE 300	0.401
12	IPE 450	0.730
13	IPE 450	0.392
14	HE 280 AA	0.262

15	IPE 450	0.739
16	HE 260 AA	0.215
17	HE 240 A	0.243
18	HE 260 AA	0.282
19	HE 260 AA	0.204
20	HE 260 AA	0.193
Frame weight	5072.94 kg	-

Table 23 shows the mean, standard deviation, minimum and maximum values of the computation time needed to run the genetic algorithm run 10 times.

Table 23. *Computation time medium frame*

	Discrete solution
Mean	7163.621 s
Standard deviation	1180.978 s
Minimum time	6086.810 s
Maximum time	8683.798 s

8.2.3 Large frame

As in the first formulation, any of the continuous algorithms is able to find a feasible solution for this reason only the discrete part had been solved.

In this case, the genetic algorithm also uses all the discrete profiles available to find the optimum solution.

Discrete

The genetic algorithm has been run 3 times and the best result for the discrete part is shown in Table 24. For this case, the mutation ratio was changed to 0.5 % as it was done in the first formulation too.

Table 24. *Discrete profiles for the big frame*

Member	Discrete solution	Member	Discrete solution
0	HE 340 AA	36	IPE 550
1	HE 400 AA	37	IPE 600
2	HE 240 A	38	IPE 450
3	HE 320 AA	39	IPE 400
4	HE 320 AA	40	HE 340 AA
5	HE 200 B	41	HE 400 AA
6	IPE 400	42	HE 340 AA
7	IPE 450	43	HE 360 AA
8	HE 340 B	44	HE 400 AA
9	HE 400 A	45	IPE 500
10	HE 280 B	46	IPE 550
11	HE 400 AA	47	IPE 500
12	HE 340 AA	48	HE 360 AA
13	HE 280 AA	49	HE 360 AA
14	HE 280 AA	50	HE 360 AA
15	IPE 330	51	HE 400 AA
16	HE 700 AA	52	IPE 500
17	HE 320 B	53	IPE 550
18	HE 280 B	54	HE 700 AA
19	HE 300 A	55	HE 800 AA
20	HE 360 AA	56	HE 450 AA
21	HE 240 A	57	IPE 550
22	HE 160 C	58	HE 360 AA
23	IPE 330	59	IPE 500
24	HE 340 B	60	IPE 500
25	HE 700 AA	61	IPE 550
26	HE 340 A	62	IPE 600
27	HE 400 AA	63	HE 650 AA
28	HE 400 AA	64	HE 320 AA
29	HE 280 AA	65	HE 800 AA
30	IPE 550	66	IPE 550
31	IPE 550	67	IPE 550
32	HE 280 A	68	IPE 600
33	HE 400 AA	69	HE 700 AA
34	HE 800 AA	70	HE 550 AA
35	IPE 500	71	IPE 500
Frame weight	32908.70 kg	-	-

Table 25 shows the mean, standard deviation, minimum and maximum values of the computation time needed to run the genetic algorithm 3 times.

Table 25. *Computation time large frame*

	Discrete solution
Mean	56334.404 s
Standard deviation	1713.602 s
Minimum time	54010.228 s
Maximum time	58090.883 s

9. DISCUSSION

9.1 Continuous part algorithms comparison

Comparing the results obtained in the continuous part with the three different algorithms, it is observed that for small frames both SLSQP and PSO are able to find the global optimum when using the first formulation in contrast to COBYLA that finds local optimums. However, PSO requires more computation time than SLSQP and for the second formulation it does not reach the best solution, see Table 26 and Table 27.

Table 26. *Small frame weight optimization results comparison*

Formulation	SLSQP	COBYLA	PSO
First	1002.12 kg	1263.63 kg	1002.12 kg
Second	789.29 kg	848.20 kg	1358.85 kg

Table 27. *Small frame computation time comparison*

Formulation	SLSQP	COBYLA	PSO
First	4.707 s	20.488 s	215.490 s
Second	82.957 s	22.091 s	988.285 s

For the medium frame, the PSO algorithm is not able to find a feasible solution and comparing SLSQP and COBYLA the first finds better results although not all the time finds the best solution, see Table 28. Comparing the computation time, COBYLA is much faster but as it does not find the global optimum is better to use SLSQP algorithm, see Table 29.

Table 28. *Medium frame weight optimization results comparison*

Formulation	SLSQP	COBYLA
First	5545.08 kg	8666.26 kg
Second	3838.01 kg	6497.96 kg

Table 29. *Medium frame computation time comparison*

Formulation	SLSQP	COBYLA
First	196.693 s	76.737 s
Second	2040.084 s	88.233 s

9.2 Formulations comparison

As expected, when comparing the solutions obtained with the two different formulations, the results with the second are better since the frame members can be from any family but it requires more computation time as there are more design variables.

For the continuous solution, the weight is quite lower with the second formulation, it is because it is finding the best results for each cross-section parameter but the relation between them does not follow any profile family correlation so when finding the discrete solution, all the commercial profiles are farther away from the optimum continuous solution. The algorithm used for the continuous solution is SLSQP which has been found to be more reliable.

However, the weight diminution for the discrete solution of the small frame is almost insignificant while it requires more time to solve the continuous part, so it is not worth it to run the second formulation for this case, see Table 30 and Table 31.

Table 30. Small frame weight optimization results comparison

Formulation	Continuous	Discrete
First	1002.12 kg	1041.89 kg
Second	789.29 kg	1036.09 kg

Table 31. *Small frame computation time comparison*

Formulation	Continuous	Discrete	Total time
First	4.707 s	1654.955 s	1659.662 s
Second	82.957 s	1621.820 s	1704.777 s

For the medium frame, the results are significantly better with the second formulation although it needs 10 times longer to solve the continuous part, the weight with the discrete formulation is a 20% higher than with the second one, view Table 32 and Table 33.

Table 32. *Medium frame weight optimization results comparison*

Formulation	Continuous	Discrete
First	5545.08 kg	6095.30kg
Second	3838.01 kg	5072.94 kg

Table 33. *Medium frame computation time comparison*

Formulation	Continuous	Discrete	Total time
First	196.693 s	6162.094 s	6358.787 s
Second	2040.084 s	7163.621 s	9203.705 s

The medium frame was also run with a mixed integer linear programming (MILP) algorithm by van Mellaert et al [51] and the results obtained were a discrete weight of 6131.87 kg in 19249 s (5.3 h) and both formulations developed in this thesis improved this result. The first formulation found a discrete weight of 6095.30 kg in 6359 s (1.7 h) and the second formulation got 5072.94 kg in 9204 s (2.5 h). This weight improvement is also because in the van Mellaert et al frame, the distribution of profiles was enforced to be symmetric and all profiles were HEA.

The big frame was solved only using the discrete part since the continuous algorithm were not able to find a solution. For this case the weight obtained with the two different formulations is a 5% more weight with the first one, but as it is a big structure it means a difference of 1790 kg which is quite significant, view Table 34. As expected, the computation time with the second formulation is longer than with the firs one due to the 5 design variables per member, view Table 35.

Table 34. *Big frame weight optimization results comparison*

Formulation	Discrete
First	34698.91 kg
Second	32908.7 kg

Table 35. *Big frame computation time comparison*

Formulation	Discrete
First	37476.553 s
Second	56334.404 s

10. CONCLUSIONS

The main purpose of this thesis was to develop a program able to solve steel frame optimization problems using a two-phase approach, satisfying all the strength and stability criteria establish in Eurocode 3, and see if it was an efficient option.

A secondary purpose was to check which algorithm could find a better solution in less computation time for the continuous part of the problem.

The last purpose was to see if it was better to establish a fix profile family before doing the optimization, this way there was only one design variable per member or if it was better to have all 5 cross-section parameters and that all members could be from any type of family.

All these questions have been solved in this thesis. First, the program developed is able to find the optimum discrete solution efficiently and with less computation time than in previous studies.

It has been found that the best algorithm to solve the continuous part is SLSQP because it is more reliable for finding global optimums than local and it can find the solution for bigger frames not like COBYLA and PSO. When the problem is a small frame PSO algorithm is also able to find the optimum solution, however, it requires more computation time than SLSQP.

Finally when comparing the two different formulations the second achieves better results because it can have all the profiles from any type of I family in contrast to the first one that has HEA columns and IPE beams. Therefore, this second formulation has 5 design variables per member instead of one and it requires more computation time.

This second formulation is useful to use it for medium and large structures where the discrete solution improves significantly even though it requires more time to solve the continuous part. However, for small frames is better to use the first formulation because both of them achieve almost the same discrete result being much faster the first one.

10.1 Further research

In the program developed, the combination of the profile type it is not enforced to be symmetric as not always the loads applied are symmetric. This symmetry has only been enforced for the second frame to be able to compare the results with the ones published by van Mellaert et al and this symmetry conditions had been set manually as constrains stablishing which frame members need to be the same.

One possible option to continue with this research is to make the program able to enforce symmetry for any kind of frame automatically instead of setting manually which members are symmetric. This way it is not necessary to buy so many different profile types.

Another possibility for continuing this research could be to implement topology optimization in combination with sizing optimization that is the one done in this thesis.

REFERENCES

[1] J. C. Maxwell, *The Scientific Papers II*, Cambridge University Press, 1890, p. 175.

[2] A. G. M. Mitchell, "The limits of economy of material in frame-structures", *the London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science*, vol.8,1904, pp. 589-597.

[3] G. N. Vanderplaats, "Structural optimization - Past. present and future" , *AIAA journal, vol. 20, 1982, pp. 992-1000.*

[4] L.A. Schmit, "Structural design by systematic synthesis"*, Proceedings of the 2nd Conference on Electronic Computation,* American Society of Civil Engineering, 1960, pp.105-122.

[5] A. J.G Schoofs, "Structural optimization history and state of the art", *Topics in Applied Mechanics, 1993,* pp. 339-345.

[6] W. Prager and R.T. Shield, "Optimum design of multi-purpose structures", *International Journal Solids and Structures*, vol. 4, 1968, pp. 469-475.

[7] W. Prager and J.E. Taylor, "Problems of optimum structural design", *Journal of Applied Mechanics,* vol. 35, 1968, pp.102-106.

[8] A.R. Toakley, "Optimum design using available sections", *Journal of the Structural Division,* vol. 94, 1968, pp. 1219-1244.

[9] S. A. Burns, *Recent Advances in optimum structural design*, ASCE Publications, 2002, pp.4-6.

[10] A. Cella and R. D. Logcher, "Automated optimum design from discrete components", *Journal of the Structural Division*, vol. 97, 1971, pp. 175-189.

[11] M. S. Voss and C. M. Foley. "Evolutionary algorithm for structural optimization", *Proceedings of the 1st Annual Conference on Genetic and Evolutionary Computation*, vol. 1, Morgan Kaufmann Publishers Inc., 1999, pp. 678-685.

[12] S. Pezeshk, C. V. Camp and D. Chen, "Design of nonlinear framed structures using genetic optimization", *Journal of structural engineering*, vol. 126, 2000, pp.382-388.

[13] C. M. Foley and D. Schinler, "Automated design of steel frames using advanced analysis and object-oriented evolutionary computation", *Journal of Structural Engineering*, vol. 129, 2003, pp. 648-660.

[14] C. V. Camp, B. J. Bichon, “Design of space trusses using ant colony optimization”, *Journal of Structural Engineering*, vol. 130, 2004, pp.741-751.

[15] C. V. Camp, B. J. Bichon and S. P. Stovall, "Design of steel frames using ant colony optimization*", Journal of Structural Engineering*, vol. 131, 2005, pp. 369-379.

[16] K. Hager and R. Balling, “New approach for discrete structural optimization”, *Journal of Structural Engineering,* vol. 114, 1988, pp. 1120-1134.

[17] J. S. Arora and M. W. Huang, Discrete structural optimization with commercially available sections, *Doboku Gakkai Ronbunshu*, vol. 1996, 1996, pp. 1-18.

[18] J. S. Arora and M. W. Huang, “Optimal design of steel structures using standard sections”, *Structural optimization*, vol. 14, 1997, pp. 24-35.

[19] C. C. Mitropoulou, Y. Fourkiotis, N. D. Lagaros and M. G. Karlaftis, “Evolution strategies-based Metaheuristic in structural design optimization”, *Metaheuristic Applications in Structures and Infrastructures*, 2013, pp. 79-102.

[20] W. Tang, L. Tong and Y. Gu, “Improved genetic algorithm for design optimization of truss structures with sizing, shape and topology variables”, *International Journal for Numerical Methods in Engineering*, vol. 62, 2005, pp. 1737-1762.

[21] S. Rojas-Labanda and M. Stople, “Benchmarking optimization solvers for structural topology optimization”, *Structural and Multidisciplinary Optimization,* vol. 52, 2015, pp. 527-547.

[22] EN 1993-1-1, *Eurocode 3: Design of steel structures-Part 1-1: General rules and rules for buildings*, CEN, 2005.

[23] A. D. Belegundu. T. R. Chandrupatla, *Optimization concepts and applications in engineering*, Prentice Hall, 1999, pp. 74-83.

[24] S.P. Bradley, A. C. Hax and T.L. Magnanti, *Applied Mathematical Programming*, Addison-Wesley Publishing Company, 1977, pp 410-464.

[25] P.R. Adby and M.A.H. Dempster, *Introduction to Optimization Methods*, Chapman and Hall, 1974, pp. 1-17.

[26] U. Kirch, *Structural optimization: Fundaments and Applications,* Springer-Verlag, 1993.

[27] R. T. Haftka and Z. Gürdal, *Elements of structural optimization*, Kluwer Academic Publishers, 1992.

[28] ArcelorMittal, *Sections and Merchant Bars Sales Programme.*

[29] M. Mckee, Nonlinear programming Chapter 11 Gradient-based nonlinear optimization methods, *Lecture delivered in Water resources systems and managment,*

Unit code: CVEN 5393. University of Colorado Boulder, 2013. Available from: http://civil.colorado.edu/~balajir/CVEN5393/

[30] Lecture 3: Gradient-based optimization. *Lecture delivered in Introduction to Multi-disciplinary Design Optimization, Unit code: AA222.* Stanford University, 2018. Available from: adl.stanford.edu/aa222/Lecture_Notes.../AA222-Lecture3.pdf

[31] Chapter 6: Gradient-Free Optimization, *Lecture delivered in Introduction to Multi-disciplinary Design Optimization, Unit code: AA222*. Stanford University, 2012, pp. 133-161. Available from: adl.stanford.edu/aa222/lecture_notes.../chapter6_grad-free.pdf

[32] W. Hare, J. Nutini, S. Tesfamariam. “A survey of non-gradient optimization methods in structural engineering”, *Advances in Engineering Software,* vol. 59, 2013, pp.19-28.

[33] R. Eberhart and J. Kennedy, "Particle swarm optimization", *Proc. IEEE International Conference on Neural Networks*, vol. 4, 1995, pp.1942-1948.

[34] R. Eberhart and J. Kennedy, "A new optimizer using particle swarm theory", *MHS'95. Proceedings of the Sixth International Symposium on Micro Machine and Human Science*, 1995, pp. 39-43.

[35] T. Zeugmann, P. Poupart, J. Kennedy, X. Jin, J. Han, L. Saitta, M. Sebag, J. Peters, J.A. Bagnell, W. Daelemans, G. I. Webb, “Particle swarm optimization”, *Encyclopedia of Machine Learning*, 2011, pp.760-766.

[36] V. Plevris, A. Batavanis and M. Papadrakakis, “Optimum design of steel structures with the particle swarm optimization method based on EC3”, *Computational Methods in Structural Dynamics and Earthquake Engineering*, 2011.

[37] K. Deb, *Multi-Objective Optimization using Evolutionary Algorithms,* John Wiley & sons Ltd, 2001.

[38] F. Streichert, *Introduction to evolutionary algorithms*, University of Tuebingen, 2002.

[39] J. Williams, *SLSQP*, 2016. [online] Available from: http://degenerateconic.com/slsqp/

[40] *SLSQP- Sequential Least Squares Programming*, 2014. [online] Available from: http://www.pyopt.org/reference/optimizers.slsqp.html

[41] M. J. D. Powell, “A direct search optimization method that models the objective and constraint function by linear interpolation”, *Advances in Optimization and Numerical Analysis*, Springer, 1994, pp. 51-67.

[42] *COBYLA-Constrained Optimization by Linear Approximation*, 2014. [online] Available from: http://www.pyopt.org/reference/optimizers.cobyla.html

[43] *scipy.optimize.minimize*. [online] Available from: https://docs.scipy.org/doc/scipy/reference/generated/scipy.optimize.minimize.html

[44] *scipy.optimize.fmin_cobyla.* [online] Available from: https://docs.scipy.org/doc/scipy-0.14.0/reference/generated/scipy.optimize.fmin_cobyla.html

[45] *Particle swarm optimization (PSO) with constraint support*. [online] Available from: https://pythonhosted.org/pyswarm/

[46] A. Osyczka. *Evolutionary algorithms for single Multicriteria Design Optimization*, Physica-Verlag, 2002.

[47] J. D. Pintér, *Global optimization: scientific and engineering case studies*. Springer Science & Business Media, 2006.

[48] A. Osyczka and K.Stanislaw, "A new constraint tournament selection method for multicriteria optimization using genetic algorithm". *Proceedings of the 2000 Congress on Evolutionary Computation,* vol. 1, 2000. Pp. 501-508.

[49] N. D. Lagaros, M. Papadrakakis and G. Kokossalakis, "Structural optimization using evolutionary algorithms", *Computer and Structures*, vol. 80, 2002, pp. 571-589.

[50] B.G.W. Craenen, A. E. Eiben and E. Marchiori, "How to handle Constraints with Evolutionary Algorithms", *Practical Handbook of Genetic Algorithms: Applications*, 2001, pp. 341-361.

[51] R. Van Mellaert, K. Mela, T. Tiainen, M. Heinisuo, G. Lombaert and M. Schevenels, "Mixed-integer linear programming approach for global discrete sizing optimization of frame structures," *Structural and Multidisciplinary Optimization*, vol. 57, 2018, pp. 579-593.

APPENDIX A. I SECTIONS DIMENSIONS

The following data includes all the cross-section parameters and other data that had been used for the calculations in the thesis and it is based on SFS-EN 10365:2017.

Table A.1. *IPE profiles*

	A **[mm^2 x10^2]**	**I_y** **[mm^4 x10^4]**	**h** **[mm]**	**b** **[mm]**	**t_w** **[mm]**	**t_f** **[mm]**	**r** **[mm]**
IPE 80	7.6	80.1	80.0	46.0	3.8	5.2	5.0
IPE 100	10.3	171.0	100.0	55.0	4.1	5.7	7.0
IPE 120	13.2	318.0	120.0	64.0	4.4	6.3	7.0
IPE 140	16.4	541.0	140.0	73.0	4.7	6.9	7.0
IPE 160	20.1	869.0	160.0	82.0	5.0	7.4	9.0
IPE 180	23.9	1317.0	180.0	91.0	5.3	8.0	9.0
IPE 200	28.5	1943.0	200.0	100.0	5.6	8.5	12.0
IPE 220	33.4	2772.0	220.0	110.0	5.9	9.2	12.0
IPE 240	39.1	3892.0	240.0	120.0	6.2	9.8	15.0
IPE 270	45.9	5790.0	270.0	135.0	6.6	10.2	15.0
IPE 300	53.8	8356.0	300.0	150.0	7.1	10.7	15.0
IPE 330	62.6	11770.0	330.0	160.0	7.5	11.5	18.0
IPE 360	72.7	16270.0	360.0	170.0	8.0	12.7	18.0
IPE 400	84.5	23130.0	400.0	180.0	8.6	13.5	21.0
IPE 450	98.8	33740.0	450.0	190.0	9.4	14.6	21.0
IPE 500	115.5	48200.0	500.0	200.0	10.2	16.0	21.0
IPE 550	134.4	67120.0	550.0	210.0	11.1	17.2	24.0
IPE 600	156.0	92080.0	600.0	220.0	12.0	19.0	24.0

Table A.2. *HEAA profiles*

	A **[mm^2 x10^2]**	**I_y** **[mm^4 x10^4]**	**h** **[mm]**	**b** **[mm]**	**t_w** **[mm]**	**t_f** **[mm]**	**r** **[mm]**
HE 100 AA	15.6	236.5	91.0	100.0	4.2	5.5	12.0
HE 120 AA	18.6	413.4	109.0	120.0	4.2	5.5	12.0
HE 140 AA	23.0	719.5	128.0	140.0	4.3	6.0	12.0
HE 160 AA	30.4	1283.0	148.0	160.0	4.5	7.0	15.0
HE 180 AA	36.5	1967.0	167.0	180.0	5.0	7.5	15.0
HE 200 AA	44.1	2944.0	186.0	200.0	5.5	8.0	18.0
HE 220 AA	51.5	4170.0	205.0	220.0	6.0	8.5	18.0
HE 240 AA	60.4	5835.0	224.0	240.0	6.5	9.0	21.0

HE 260 AA	69.0	7981.0	244.0	260.0	6.5	9.5	24.0
HE 280 AA	78.0	10560.0	264.0	280.0	7.0	10.0	24.0
HE 300 AA	88.9	13800.0	283.0	300.0	7.5	10.5	27.0
HE 320 AA	94.6	16450.0	301.0	300.0	8.0	11.0	27.0
HE 340 AA	100.5	19550.0	320.0	300.0	8.5	11.5	27.0
HE 360 AA	106.6	23040.0	339.0	300.0	9.0	12.0	27.0
HE 400 AA	117.7	31250.0	378.0	300.0	9.5	13.0	27.0
HE 450 AA	127.1	41890.0	425.0	300.0	10.0	13.5	27.0
HE 500 AA	136.9	54640.0	472.0	300.0	10.5	14.0	27.0
HE 550 AA	152.8	72870.0	522.0	300.0	11.5	15.0	27.0
HE 600 AA	164.1	91900.0	571.0	300.0	12.0	15.5	27.0
HE 650 AA	175.8	113900.0	620.0	300.0	12.5	16.0	27.0
HE 700 AA	190.9	142700.0	670.0	300.0	13.0	17.0	27.0
HE 800 AA	218.5	208900.0	770.0	300.0	14.0	18.0	30.0
HE 900 AA	252.2	301100.0	870.0	300.0	15.0	20.0	30.0
HE 1000 AA	282.2	406500.0	970.0	300.0	16.0	21.0	30.0

Table A.3. *HEA profiles*

	A **[mm^2 x10^2]**	**I_y** **[mm^4 x10^4]**	**h** **[mm]**	**b** **[mm]**	**t_w** **[mm]**	**t_f** **[mm]**	**r** **[mm]**
HE 100 A	21.2	349.2	96.0	100.0	5.0	8.0	12.0
HE 120 A	25.3	606.2	114.0	120.0	5.0	8.0	12.0
HE 140 A	31.4	1033.0	133.0	140.0	5.5	8.5	12.0
HE 160 A	38.8	1673.0	152.0	160.0	6.0	9.0	15.0
HE 180 A	45.3	2510.0	171.0	180.0	6.0	9.5	15.0
HE 200 A	53.8	3692.0	190.0	200.0	6.5	10.0	18.0
HE 220 A	64.3	5410.0	210.0	220.0	7.0	11.0	18.0
HE 240 A	76.8	7763.0	230.0	240.0	7.5	12.0	21.0
HE 260 A	86.8	10450.0	250.0	260.0	7.5	12.5	24.0
HE 280 A	97.3	13670.0	270.0	280.0	8.0	13.0	24.0
HE 300 A	112.5	18260.0	290.0	300.0	8.5	14.0	27.0
HE 320 A	124.4	22930.0	310.0	300.0	9.0	15.5	27.0
HE 340 A	133.5	27690.0	330.0	300.0	9.5	16.5	27.0
HE 360 A	142.8	33090.0	350.0	300.0	10.0	17.5	27.0
HE 400 A	159.0	45070.0	390.0	300.0	11.0	19.0	27.0

HE 450 A	178.0	63720.0	440.0	300.0	11.5	21.0	27.0
HE 500 A	197.5	86970.0	490.0	300.0	12.0	23.0	27.0
HE 550 A	211.8	111900.0	540.0	300.0	12.5	24.0	27.0
HE 600 A	226.5	141200.0	590.0	300.0	13.0	25.0	27.0
HE 650 A	241.6	175200.0	640.0	300.0	13.5	26.0	27.0
HE 700 A	260.5	215300.0	690.0	300.0	14.5	27.0	27.0
HE 800 A	285.8	303400.0	790.0	300.0	15.0	28.0	30.0
HE 900 A	320.5	422100.0	890.0	300.0	16.0	30.0	30.0
HE 1000 A	346.8	553800.0	990.0	300.0	16.5	31.0	30.0

Table A.4. *HEB profiles*

	A **[mm^2 x10^2]**	**I_y** **[mm^4 x10^4]**	**h** **[mm]**	**b** **[mm]**	**t_w** **[mm]**	**t_f** **[mm]**	**r** **[mm]**
HE 100 B	26.0	449.5	100.0	100.0	6.0	10.0	12.0
HE 120 B	34.0	864.4	120.0	120.0	6.5	11.0	12.0
HE 140 B	43.0	1509.0	140.0	140.0	7.0	12.0	12.0
HE 160 B	54.3	2492.0	160.0	160.0	8.0	13.0	15.0
HE 180 B	65.3	3831.0	180.0	180.0	8.5	14.0	15.0
HE 200 B	78.1	5696.0	200.0	200.0	9.0	15.0	18.0
HE 220 B	91.0	8091.0	220.0	220.0	9.5	16.0	18.0
HE 240 B	106.0	11260.0	240.0	240.0	10.0	17.0	21.0
HE 260 B	118.4	14920.0	260.0	260.0	10.0	17.5	24.0
HE 280 B	131.4	19270.0	280.0	280.0	10.5	18.0	24.0
HE 300 B	149.1	25170.0	300.0	300.0	11.0	19.0	27.0
HE 320 B	161.3	30820.0	320.0	300.0	11.5	20.5	27.0
HE 340 B	170.9	36660.0	340.0	300.0	12.0	21.5	27.0
HE 360 B	180.6	43190.0	360.0	300.0	12.5	22.5	27.0
HE 400 B	197.8	57680.0	400.0	300.0	13.5	24.0	27.0
HE 450 B	218.0	79890.0	450.0	300.0	14.0	26.0	27.0
HE 500 B	238.6	107200.0	500.0	300.0	14.5	28.0	27.0
HE 550 B	254.1	136700.0	550.0	300.0	15.0	29.0	27.0
HE 600 B	270.0	171000.0	600.0	300.0	15.5	30.0	27.0
HE 650 B	286.3	210600.0	650.0	300.0	16.0	31.0	27.0
HE 700 B	306.4	256900.0	700.0	300.0	17.0	32.0	27.0
HE 800 B	334.2	359100.0	800.0	300.0	17.5	33.0	30.0

HE 900 B	371.3	494100.0	900.0	300.0	18.5	35.0	30.0
HE 1000 B	400.0	644700.0	1000.0	300.0	19.0	36.0	30.0

Table A.5. *HEC profiles*

	A [mm^2 x10^2]	**I_y [mm^4 x10^4]**	**h [mm]**	**b [mm]**	**t_w [mm]**	**t_f [mm]**	**r [mm]**
HE 100 C	39.3	758.7	110.0	103.0	9.0	15.0	12.0
HE 120 C	49.9	1388.0	130.0	123.0	9.5	16.0	12.0
HE 140 C	61.5	2330.0	150.0	143.0	10.0	17.0	12.0
HE 160 C	75.4	3704.0	170.0	163.0	11.0	18.0	15.0
HE 180 C	89.0	5543.0	190.0	183.0	11.5	19.0	15.0
HE 200 C	104.4	8029.0	210.0	203.0	12.0	20.0	18.0
HE 220 C	119.9	11180.0	230.0	223.0	12.5	21.0	18.0
HE 240 C	152.2	17330.0	255.0	244.0	14.0	24.5	21.0
HE 260 C	168.4	22590.0	275.0	264.0	14.0	25.0	24.0
HE 280 C	185.2	28810.0	295.0	284.0	14.5	25.5	24.0
HE 300 C	225.1	40950.0	320.0	305.0	16.0	29.0	27.0
HE 320 C	236.9	48710.0	340.0	305.0	16.0	30.5	27.0

Table A.6. *HEM profiles*

	A [mm^2 x10^2]	**I_y [mm^4 x10^4]**	**h [mm]**	**b [mm]**	**t_w [mm]**	**t_f [mm]**	**r [mm]**
HE 100 M	53.2	1143.0	120.0	106.0	12.0	20.0	12.0
HE 120 M	66.4	2018.0	140.0	126.0	12.5	21.0	12.0
HE 140 M	80.6	3291.0	160.0	146.0	13.0	22.0	12.0
HE 160 M	97.1	5098.0	180.0	166.0	14.0	23.0	15.0
HE 180 M	113.3	7483.0	200.0	186.0	14.5	24.0	15.0
HE 200 M	131.3	10640.0	220.0	206.0	15.0	25.0	18.0
HE 220 M	149.4	14600.0	240.0	226.0	15.5	26.0	18.0
HE 240 M	199.6	24290.0	270.0	248.0	18.0	32.0	21.0
HE 260 M	219.6	31310.0	290.0	268.0	18.0	32.5	24.0
HE 280 M	240.2	39550.0	310.0	288.0	18.5	33.0	24.0
HE 300 M	303.1	59200.0	340.0	310.0	21.0	39.0	27.0
HE 320 M	312.0	68130.0	359.0	309.0	21.0	40.0	27.0

HE 340 M	315.8	76370.0	377.0	309.0	21.0	40.0	27.0
HE 360 M	318.8	84870.0	395.0	308.0	21.0	40.0	27.0
HE 400 M	325.8	104100.0	432.0	307.0	21.0	40.0	27.0
HE 450 M	335.4	131500.0	478.0	307.0	21.0	40.0	27.0
HE 500 M	344.3	161900.0	524.0	306.0	21.0	40.0	27.0
HE 550 M	354.4	198000.0	572.0	306.0	21.0	40.0	27.0
HE 600 M	363.7	237400.0	620.0	305.0	21.0	40.0	27.0
HE 650 M	373.7	281700.0	668.0	305.0	21.0	40.0	27.0
HE 700 M	383.0	329300.0	716.0	304.0	21.0	40.0	27.0
HE 800 M	404.3	442600.0	814.0	303.0	21.0	40.0	30.0
HE 900 M	423.6	570400.0	910.0	302.0	21.0	40.0	30.0
HE 1000 M	444.2	722300.0	1008.0	302.0	21.0	40.0	30.0